



PCI Express® Base Specification Revision 6.4

June 05, 2025

Copyright© 2002-2025 PCI-SIG

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.

This PCI Specification is provided “as is” without any warranties of any kind, including any warranty of merchantability, non-infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. PCI-SIG disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. This document itself may not be modified in any way, including by removing the copyright notice or references to PCI-SIG. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein. PCI, PCI Express, PCIe, and PCI-SIG are trademarks or registered trademarks of PCI-SIG. All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Base 6.4 vs Base 6.3

Table of Contents

| | | |
|-----------|--|------------|
| 1. | Introduction..... | 141 |
| 1.1 | An Evolving I/O Interconnect | 141 |
| 1.2 | PCI Express Link..... | 142 |
| 1.3 | PCI Express Fabric Topology | 143 |
| 1.3.1 | Root Complex..... | 144 |
| 1.3.2 | Endpoints | 145 |
| 1.3.2.1 | Legacy Endpoint Rules..... | 145 |
| 1.3.2.2 | PCI Express Endpoint Rules | 146 |
| 1.3.2.3 | Root Complex Integrated Endpoint Rules..... | 146 |
| 1.3.3 | Switch | 147 |
| 1.3.4 | Root Complex Event Collector..... | 148 |
| 1.3.5 | PCI Express to PCI/PCI-X Bridge..... | 148 |
| 1.4 | Hardware/Software Model for Discovery, Configuration and Operation | 148 |
| 1.5 | PCI Express Layering Overview..... | 149 |
| 1.5.1 | Transaction Layer..... | 150 |
| 1.5.2 | Data Link Layer | 150 |
| 1.5.3 | Physical Layer..... | 150 |
| 1.5.4 | Layer Functions and Services | 151 |
| 1.5.4.1 | Transaction Layer Services | 151 |
| 1.5.4.2 | Data Link Layer Services | 152 |
| 1.5.4.3 | Physical Layer Services | 152 |
| 1.5.4.4 | Inter-Layer Interfaces | 153 |
| 1.5.4.4.1 | Transaction/Data Link Interface | 153 |
| 1.5.4.4.2 | Data Link/Physical Interface | 153 |
| 2. | Transaction Layer Specification | 155 |
| 2.1 | Transaction Layer Overview | 155 |
| 2.1.1 | Address Spaces, Transaction Types, and Usage | 155 |
| 2.1.1.1 | Memory Transactions..... | 156 |
| 2.1.1.2 | I/O Transactions | 156 |
| 2.1.1.3 | Configuration Transactions | 157 |
| 2.1.1.4 | Message Transactions | 157 |
| 2.1.2 | Packet Format Overview..... | 157 |
| 2.2 | Transaction Layer Protocol - Packet Definition..... | 159 |
| 2.2.1 | Common Packet Header Fields | 159 |
| 2.2.1.1 | Common Packet Header Fields for Non-Flit Mode | 159 |
| 2.2.1.2 | Common Packet Header Fields for Flit Mode..... | 162 |
| 2.2.2 | TLPs with Data Payloads - Rules..... | 186 |
| 2.2.3 | TLP Digest Rules - Non-Flit Mode Only | 189 |
| 2.2.4 | Routing and Addressing Rules..... | 190 |
| 2.2.4.1 | ↑↓Address Based↓ ↑↑Address Based↑ Routing Rules | 190 |
| 2.2.4.2 | ID Based Routing Rules | 197 |
| 2.2.5 | First/Last DW Byte Enables Rules | 207 |
| 2.2.5.1 | Byte Enable Rules for Non-Flit Mode..... | 207 |
| 2.2.5.2 | Byte Enable Rules for Flit Mode..... | 210 |
| 2.2.6 | Transaction Descriptor..... | 211 |

Base 6.4 vs Base 6.3

| | | |
|------------|--|-----------------------------|
| 2.2.6.1 | Overview | 211 |
| 2.2.6.2 | Transaction Descriptor - Transaction ID Field | 211 |
| 2.2.6.2.1 | ↑↑Tag Rules for Non-UIO Channel Operation↑ | 212 |
| 2.2.6.2.2 | ↑↑Tag Rules for UIO Channel Operation↑ | 219 |
| | | Errata: Base 6.3 B807△◀▷ |
| | | Errata: Base 6.3 B807△◀▷ |
| 2.2.6.3 | Transaction Descriptor - Attributes Field | 220 |
| 2.2.6.4 | Relaxed Ordering and ID-Based Ordering Attributes | 220 |
| 2.2.6.5 | No Snoop Attribute | 221 |
| 2.2.6.6 | Transaction Descriptor - Traffic Class Field | 221 |
| 2.2.7 | Memory, I/O, and Configuration Request Rules..... | 222 |
| 2.2.7.1 | Non-Flit Mode | 222 |
| 2.2.7.1.1 | TPH Rules ↑↑- Non-Flit Mode↑ | 227 |
| 2.2.7.2 | Flit Mode | 232 |
| 2.2.8 | Message Request Rules | 234 |
| 2.2.8.1 | INTx Interrupt Signaling - Rules | 238 |
| 2.2.8.2 | Power Management Messages | 241 |
| 2.2.8.3 | Error Signaling Messages | 242 |
| 2.2.8.4 | Locked Transactions Support | 244 |
| 2.2.8.5 | Slot Power Limit Support | 245 |
| 2.2.8.6 | Vendor-Defined Messages | 246 |
| 2.2.8.6.1 | PCI-SIG Defined VDMs | 248 |
| 2.2.8.6.2 | Device Readiness Status (DRS) Message | 249 |
| 2.2.8.6.3 | Function Readiness Status Message (FRS Message) | 250 |
| 2.2.8.6.4 | Hierarchy ID Message | 252 |
| 2.2.8.7 | Ignored Messages | 254 |
| 2.2.8.8 | Latency Tolerance Reporting (LTR) Message | 255 |
| 2.2.8.9 | Optimized Buffer Flush/Fill (OBFF) Message | 256 |
| 2.2.8.10 | Precision Time Measurement (PTM) Messages | 258 |
| 2.2.8.11 | Integrity and Data Encryption (IDE) Messages | 261 |
| 2.2.9 | Completion Rules | 267 |
| 2.2.9.1 | Completion Rules for Non-Flit Mode | 267 |
| 2.2.9.2 | Completion Rules for Flit Mode | 271 |
| 2.2.10 | TLP Prefix Rules | 275 |
| 2.2.10.1 | TLP Prefix General Rules - Non-Flit Mode | 275 |
| 2.2.10.2 | Local TLP Prefix Processing | 275 |
| 2.2.10.2.1 | Vendor Defined Local TLP Prefix | 276 |
| 2.2.10.3 | Flit Mode Local TLP Prefix | 276 |
| 2.2.10.4 | End-End TLP Prefix Processing - Non-Flit Mode | 277 |
| 2.2.10.4.1 | Vendor Defined End-End TLP Prefix | 278 |
| 2.2.10.4.2 | Root Ports with ↑↑End-End TLP Prefix Supported↓ ↑↑End-End TLP Prefix Supported↑ | 278 |
| 2.2.11 | OHC-E Rules - Flit Mode | 279 |
| 2.3 | Handling of Received TLPs | 281 |
| 2.3.1 | Request Handling Rules | 284 |
| 2.3.1.1 | Data Return for Non-UIO Read Requests | 290 |
| 2.3.1.2 | UIO Read Completions | 295 |
| 2.3.1.3 | UIO Write Completions | 296 |
| 2.3.2 | Completion Handling Rules | 296 |
| 2.4 | Transaction Ordering | 298 |
| 2.4.1 | Transaction Ordering Rules for TLPs not using UIO or Flow-Through IDE Streams | 298 |

| | | |
|-----------|---|-----|
| 2.4.2 | Ordering Rules for UIO | 304 |
| 2.4.3 | Update Ordering and Granularity Observed by a Read Transaction | 305 |
| 2.4.3.1 | Ordering and Granularity for Non-UIO Reads | 305 |
| 2.4.3.2 | Ordering and Granularity for UIO Reads | 306 |
| 2.4.4 | Update Ordering and Granularity Provided by a Write Transaction | 306 |
| 2.4.4.1 | Ordering and Granularity for Non-UIO Writes | 306 |
| 2.4.4.2 | Ordering and Granularity for UIO Writes | 306 |
| 2.5 | Virtual Channel (VC) Mechanism | 307 |
| 2.5.1 | Virtual Channel Identification (VC ID) | 311 |
| 2.5.2 | TC to VC Mapping | 311 |
| 2.5.3 | VC and TC Rules | 313 |
| 2.6 | Ordering and Receive Buffer Flow Control | 314 |
| 2.6.1 | Flow Control (FC) Rules | 315 |
| 2.6.1.1 | FC Information Tracked by Transmitter | 323 |
| 2.6.1.2 | FC Information Tracked by Receiver | 329 |
| 2.7 | End-to-End Data Integrity | 336 |
| 2.7.1 | ECRC Rules | 336 |
| 2.7.2 | Error Forwarding (Data Poisoning) | 341 |
| 2.7.2.1 | Rules For Use of Data Poisoning | 342 |
| 2.8 | Completion Timeout Mechanism | 343 |
| 2.9 | Link Status Dependencies | 344 |
| 2.9.1 | Transaction Layer Behavior in DL_Down Status | 344 |
| 2.9.2 | Transaction Layer Behavior in DL_Up Status | 345 |
| 2.9.3 | Transaction Layer Behavior During Downstream Port Containment | 346 |
| 3. | Data Link Layer Specification | 349 |
| 3.1 | Data Link Layer Overview | 349 |
| 3.2 | Data Link Control and Management State Machine | 350 |
| 3.2.1 | Data Link Control and Management State Machine Rules | 351 |
| 3.3 | Data Link Feature Exchange | 354 |
| 3.4 | Flow Control Initialization Protocol | 356 |
| 3.4.1 | Flow Control Initialization State Machine Rules | 356 |
| 3.4.2 | Scaled Flow Control | 364 |
| 3.5 | Data Link Layer Packets (DLLPs) | 364 |
| 3.5.1 | Data Link Layer Packet Rules | 365 |
| 3.6 | Data Integrity Mechanisms | 376 |
| 3.6.1 | Introduction | 376 |
| 3.6.2 | LCRC, ↑↑TLP↑ Sequence Number, and Retry Management (TLP Transmitter) | 377 |
| 3.6.2.1 | LCRC and ↑↑TLP↑ Sequence Number Rules (TLP Transmitter) | 377 |
| 3.6.2.2 | Handling of Received DLLPs (Non-Flit Mode) | 385 |
| 3.6.2.3 | Handling of Received DLLPs (Flit Mode) | 386 |
| 3.6.3 | LCRC and ↑↑TLP↑ Sequence Number (TLP Receiver) (Non-Flit Mode) | 387 |
| 3.6.3.1 | LCRC and ↑↑TLP↑ Sequence Number Rules (TLP Receiver) | 387 |
| 4. | Physical Layer Logical Block | 393 |
| 4.1 | Introduction | 393 |
| 4.2 | Logical Sub-block | 393 |
| 4.2.1 | 8b/10b Encoding for 2.5 GT/s and 5.0 GT/s Data Rates | 395 |
| 4.2.1.1 | Symbol Encoding | 395 |
| 4.2.1.1.1 | Serialization and De-serialization of Data | 396 |
| 4.2.1.1.2 | Special Symbols for Framing and Link Management (K Codes) | 397 |

| | | |
|---------------|---|-----|
| 4.2.1.1.3 | 8b/10b Decode Rules | 398 |
| 4.2.1.2 | Framing and Application of Symbols to Lanes | 399 |
| 4.2.1.2.1 | Framing and Application of Symbols to Lanes for TLPs and DLLPs in Non-Flit Mode | 399 |
| 4.2.1.3 | Data Scrambling..... | 402 |
| 4.2.2 | 128b/130b Encoding for 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s Data Rates | 403 |
| 4.2.2.1 | Lane Level Encoding | 404 |
| 4.2.2.2 | Ordered Set Blocks..... | 405 |
| 4.2.2.2.1 | Block Alignment..... | 405 |
| 4.2.2.3 | Data Blocks..... | 406 |
| 4.2.2.3.1 | Framing Tokens in Non-Flit Mode Non-Flit Mode | 406 |
| 4.2.2.3.2 | Transmitter Framing Requirements in Non-Flit Mode..... | 411 |
| 4.2.2.3.3 | Receiver Framing Requirements in Non-Flit Mode | 412 |
| 4.2.2.3.4 | Receiver Framing Requirements in Flit Mode | 414 |
| 4.2.2.3.5 | Recovery from Framing Errors in Non-Flit Mode and Flit Mode | 415 |
| 4.2.2.4 | Scrambling in Non-Flit Mode and Flit Mode | 415 |
| 4.2.2.5 | Precoding..... | 421 |
| 4.2.2.5.1 | Precoding at 32.0 GT/s Data Rate..... | 422 |
| 4.2.2.6 | Loopback with 128b/130b Code in Non-Flit Mode and Flit Mode..... | 424 |
| 4.2.3 | Flit Mode Operation | 424 |
| 4.2.3.1 | 1b/1b Encoding for 64.0 GT/s and higher Data Rates | 424 |
| 4.2.3.1.1 | PAM4 Signaling..... | 426 |
| 4.2.3.1.2 | 1b/1b Scrambling..... | 427 |
| 4.2.3.1.3 | Gray Coding at 64.0 GT/s and Higher Data Rates | 428 |
| 4.2.3.1.4 | Precoding at 64.0 GT/s and Higher Data Rates..... | 429 |
| 4.2.3.1.5 | Ordered Set Blocks at 64.0 GT/s and Higher Data Rates..... | 432 |
| 4.2.3.1.6 | Alignment at Block/ Flit Level for 1b/1b Encoding | 433 |
| 4.2.3.2 | Processing of Ordered Sets During Flit Mode Data Stream | 434 |
| 4.2.3.3 | Data Stream in Flit Mode..... | 436 |
| 4.2.3.4 | Bytes in Flit Layout..... | 442 |
| 4.2.3.4.1 | TLP Bytes in Flit | 442 |
| 4.2.3.4.2 | DLP Bytes in Flit..... | 445 |
| 4.2.3.4.2.1 | Flit Sequence Number and Retry Mechanism | 449 |
| 4.2.3.4.2.1.1 | IDLE Flit Handshake Phase | 456 |
| 4.2.3.4.2.1.2 | Sequence Number Handshake Phase..... | 457 |
| 4.2.3.4.2.1.3 | Normal Flit Exchange Phase | 458 |
| 4.2.3.4.2.1.4 | Received Ack and Nak Processing | 460 |
| 4.2.3.4.2.1.5 | Ack, Nak, and Discard Rules | 461 |
| 4.2.3.4.2.1.6 | Flit Replay Scheduling | 468 |
| 4.2.3.4.2.1.7 | Flit Replay Transmit Rules | 469 |
| 4.2.3.4.3 | NOP Flit Payload..... | 476 |
| 4.2.3.4.3.1 | NOP.Empty Flit | 476 |
| 4.2.3.4.3.2 | NOP.Debug Flit | 477 |
| 4.2.3.4.3.2.1 | PCI-SIG Defined Debug Chunk Opcode Values | 481 |
| 4.2.3.4.3.2.2 | Empty Debug Chunk | 482 |
| 4.2.3.4.3.2.3 | Start Capture Trigger Debug Chunk | 483 |
| 4.2.3.4.3.2.4 | Stop Capture Trigger Debug Chunk..... | 484 |
| 4.2.3.4.3.2.5 | FC Information Tracked by Transmitter Debug Chunk | 484 |
| 4.2.3.4.3.2.6 | FC Information Tracked by Receiver Debug Chunk | 486 |
| 4.2.3.4.3.2.7 | Flit Mode Transmitter Retry Flags and Counters Debug Chunk | 488 |
| 4.2.3.4.3.2.8 | Flit Mode Receiver Retry Flags and Counters Debug Chunk | 490 |
| 4.2.3.4.3.2.9 | Buffer Occupancy Debug Chunk | 491 |

| | | |
|----------------|--|-----|
| 4.2.3.4.3.2.10 | Link Debug Request Debug Chunk | 492 |
| 4.2.3.4.3.3 | NOP.Vendor Flit | 493 |
| 4.2.3.4.4 | CRC Bytes in Flit | 494 |
| 4.2.3.4.5 | ECC Bytes in Flit | 494 |
| 4.2.3.4.6 | Ordered Set insertion in Data Stream in Flit Mode | 502 |
| 4.2.4 | Link Equalization Procedure for 8.0 GT/s and Higher Data Rates | 503 |
| 4.2.4.1 | Rules for Transmitter Coefficients | 515 |
| 4.2.4.2 | Encoding of Presets..... | 516 |
| 4.2.5 | Link Initialization and Training | 517 |
| 4.2.5.1 | Training Sequences..... | 518 |
| 4.2.5.2 | Alternate Protocol Negotiation..... | 539 |
| 4.2.5.3 | Electrical Idle Sequences (EIOS and EIEOS) | 542 |
| 4.2.5.4 | Inferring Electrical Idle..... | 547 |
| 4.2.5.5 | Lane Polarity Inversion | 548 |
| 4.2.5.6 | Fast Training Sequence (FTS) | 549 |
| 4.2.5.7 | Start of Data Stream Ordered Set (SDS Ordered Set)..... | 551 |
| 4.2.5.8 | Link Error Recovery..... | 551 |
| 4.2.5.9 | Reset | 552 |
| 4.2.5.9.1 | Fundamental Reset | 552 |
| 4.2.5.9.2 | Hot Reset | 553 |
| 4.2.5.10 | Link Data Rate Negotiation | 553 |
| 4.2.5.11 | Link Width and Lane Sequence Negotiation..... | 553 |
| 4.2.5.11.1 | Required and Optional Port Behavior | 553 |
| 4.2.5.12 | Lane-to-Lane De-skew | 554 |
| 4.2.5.13 | Lane vs. Link Training..... | 555 |
| 4.2.6 | Link Training and Status State Machine (LTSSM) Descriptions | 555 |
| 4.2.6.1 | Detect Overview | 556 |
| 4.2.6.2 | Polling Overview | 556 |
| 4.2.6.3 | Configuration Overview | 556 |
| 4.2.6.4 | Recovery Overview..... | 556 |
| 4.2.6.5 | L0 Overview | 557 |
| 4.2.6.6 | L0s Overview | 557 |
| 4.2.6.7 | L0p Overview..... | 557 |
| 4.2.6.7.1 | Link Management DLLP | 562 |
| 4.2.6.8 | L1 Overview | 565 |
| 4.2.6.9 | L2 Overview | 565 |
| 4.2.6.10 | Disabled Overview | 566 |
| 4.2.6.11 | Loopback Overview | 566 |
| 4.2.6.12 | Hot Reset Overview..... | 567 |
| 4.2.7 | Link Training and Status State Rules | 567 |
| 4.2.7.1 | Detect..... | 569 |
| 4.2.7.1.1 | Detect.Quiet | 569 |
| 4.2.7.1.2 | Detect.Active | 570 |
| 4.2.7.2 | Polling..... | 571 |
| 4.2.7.2.1 | Polling.Active..... | 571 |
| 4.2.7.2.2 | Polling.Compliance..... | 573 |
| 4.2.7.2.3 | Polling.Configuration..... | 578 |
| 4.2.7.2.4 | Polling.Speed | 578 |
| 4.2.7.3 | Configuration | 579 |
| 4.2.7.3.1 | Configuration.Linkwidth.Start..... | 579 |
| 4.2.7.3.1.1 | Downstream Lanes | 579 |

| | | |
|---------------|---|-----|
| 4.2.7.3.1.2 | Upstream Lanes | 581 |
| 4.2.7.3.2 | Configuration.Linkwidth.Accept | 583 |
| 4.2.7.3.2.1 | Downstream Lanes | 583 |
| 4.2.7.3.2.2 | Upstream Lanes | 584 |
| 4.2.7.3.3 | Configuration.Lanenum.Accept | 587 |
| 4.2.7.3.3.1 | Downstream Lanes | 588 |
| 4.2.7.3.3.2 | Upstream Lanes | 589 |
| 4.2.7.3.4 | Configuration.Lanenum.Wait | 590 |
| 4.2.7.3.4.1 | Downstream Lanes | 591 |
| 4.2.7.3.4.2 | Upstream Lanes | 591 |
| 4.2.7.3.5 | Configuration.Complete | 591 |
| 4.2.7.3.5.1 | Downstream Lanes | 592 |
| 4.2.7.3.5.2 | Upstream Lanes | 594 |
| 4.2.7.3.6 | Configuration.Idle | 595 |
| 4.2.7.4 | Recovery | 600 |
| 4.2.7.4.1 | Recovery.RcvrLock | 600 |
| 4.2.7.4.2 | Recovery.Equalization | 606 |
| 4.2.7.4.2.1 | Downstream Lanes | 607 |
| 4.2.7.4.2.1.1 | Phase 1 of Transmitter Equalization | 608 |
| 4.2.7.4.2.1.2 | Phase 2 of Transmitter Equalization | 610 |
| 4.2.7.4.2.1.3 | Phase 3 of Transmitter Equalization | 611 |
| 4.2.7.4.2.2 | Upstream Lanes | 613 |
| 4.2.7.4.2.2.1 | Phase 0 of Transmitter Equalization | 614 |
| 4.2.7.4.2.2.2 | Phase 1 of Transmitter Equalization | 616 |
| 4.2.7.4.2.2.3 | Phase 2 of Transmitter Equalization | 616 |
| 4.2.7.4.2.2.4 | Phase 3 of Transmitter Equalization | 619 |
| 4.2.7.4.3 | Recovery.Speed | 620 |
| 4.2.7.4.4 | Recovery.RcvrCfg | 621 |
| 4.2.7.4.5 | Recovery.Idle | 628 |
| 4.2.7.5 | L0 | 631 |
| 4.2.7.6 | L0s | 633 |
| 4.2.7.6.1 | Receiver L0s | 633 |
| 4.2.7.6.1.1 | Rx_L0s.Entry | 633 |
| 4.2.7.6.1.2 | Rx_L0s.Idle | 633 |
| 4.2.7.6.1.3 | Rx_L0s.FTS | 634 |
| 4.2.7.6.2 | Transmitter L0s | 634 |
| 4.2.7.6.2.1 | Tx_L0s.Entry | 634 |
| 4.2.7.6.2.2 | Tx_L0s.Idle | 635 |
| 4.2.7.6.2.3 | Tx_L0s.FTS | 635 |
| 4.2.7.7 | L1 | 636 |
| 4.2.7.7.1 | L1.Entry | 636 |
| 4.2.7.7.2 | L1.Idle | 636 |
| 4.2.7.8 | L2 | 638 |
| 4.2.7.8.1 | L2.Idle | 638 |
| 4.2.7.8.2 | L2.TransmitWake | 639 |
| 4.2.7.9 | Disabled | 639 |
| 4.2.7.10 | Loopback | 640 |
| 4.2.7.10.1 | Loopback.Entry | 640 |
| 4.2.7.10.2 | Loopback.Active | 644 |
| 4.2.7.10.3 | Loopback.Exit | 646 |
| 4.2.7.11 | Hot Reset | 647 |

| | | |
|------------|--|----------------------------|
| 4.2.8 | Clock Tolerance Compensation | 648 |
| 4.2.8.1 | SKP Ordered Set for 8b/10b Encoding | 649 |
| 4.2.8.2 | SKP Ordered Set for 128b/130b Encoding | 649 |
| 4.2.8.3 | SKP Ordered Set for 1b/1b Encoding | 653 |
| 4.2.8.4 | Rules for Transmitters | 658 |
| 4.2.8.5 | Rules for Receivers | 661 |
| 4.2.8.6 | ↑↑SKP Ordered Set Usage with Optical Retimer Solutions↑ | 662 |
| | | ECN: Base 6.3 Optical△↔ |
| 4.2.9 | Compliance Pattern in 8b/10b Encoding | 664 |
| 4.2.10 | Modified Compliance Pattern in 8b/10b Encoding | 665 |
| 4.2.11 | Compliance Pattern in 128b/130b Encoding | 666 |
| 4.2.12 | Modified Compliance Pattern in 128b/130b Encoding | 668 |
| 4.2.13 | Jitter Measurement Pattern in 128b/130b | 669 |
| 4.2.14 | Compliance Pattern in 1b/1b Encoding | 669 |
| 4.2.15 | Modified Compliance Pattern in 1b/1b Encoding | 670 |
| 4.2.16 | Jitter Measurement Pattern in 1b/1b Encoding | 671 |
| 4.2.17 | Toggle Patterns in 1b/1b encoding | 671 |
| 4.2.18 | Lane Margining at Receiver | 672 |
| 4.2.18.1 | Receiver Number, Margin Type, Usage Model, and Margin Payload Fields | 672 |
| 4.2.18.1.1 | Step Margin Execution Status | 678 |
| 4.2.18.1.2 | Margin Payload for Step Margin Commands | 678 |
| 4.2.18.2 | Margin Command and Response Flow | 679 |
| 4.2.18.3 | Flit Mode 8.0 GT/s Margining Behavior | 682 |
| 4.2.18.4 | Receiver Margin Testing Requirements | 682 |
| 4.3 | Retimers | 687 |
| 4.3.1 | Retimer Requirements | 688 |
| 4.3.2 | Supported Retimer Topologies | 689 |
| 4.3.3 | Variables | 692 |
| 4.3.4 | Receiver Impedance Propagation Rules | 693 |
| 4.3.5 | Switching Between Modes | 693 |
| 4.3.6 | ↑↑Activation Rules↑ | 693 |
| | | ECN: Base 6.3 Optical△↔ |
| 4.3.7 | Forwarding Rules | 697 |
| 4.3.7.1 | Forwarding Type Rules | 700 |
| 4.3.7.2 | Orientation, Lane Numbers, and Data Stream Mode Rules | 700 |
| 4.3.7.3 | Electrical Idle Exit Rules | 702 |
| 4.3.7.4 | Data Rate Change and Determination Rules | 705 |
| 4.3.7.5 | Electrical Idle Entry Rules | 705 |
| 4.3.7.6 | Transmitter Settings Determination Rules | 707 |
| 4.3.7.7 | Ordered Set Modification Rules | 709 |
| 4.3.7.8 | DLLP, TLP, Logical Idle, and Flit Modification Rules | 712 |
| 4.3.7.9 | 8b/10b Encoding Rules | 712 |
| 4.3.7.10 | 8b/10b Scrambling Rules | 713 |
| 4.3.7.11 | Hot Reset Rules | 713 |
| 4.3.7.12 | Disable Link Rules | 713 |
| 4.3.7.13 | Loopback | 714 |
| 4.3.7.14 | Compliance Receive Rules | 715 |
| 4.3.7.15 | Enter Compliance Rules | 716 |
| 4.3.8 | Execution Mode Rules | 719 |
| 4.3.8.1 | CompLoadBoard Rules | 719 |

| | | |
|-------------|---|-----|
| 4.3.8.1.1 | CompLoadBoard.Entry | 720 |
| 4.3.8.1.2 | CompLoadBoard.Pattern | 720 |
| 4.3.8.1.3 | CompLoadBoard.Exit | 721 |
| 4.3.8.2 | Link Equalization Rules | 721 |
| 4.3.8.2.1 | Downstream Lanes | 722 |
| 4.3.8.2.1.1 | Phase 1 | 722 |
| 4.3.8.2.1.2 | Phase 2 | 722 |
| 4.3.8.2.1.3 | Phase 3 Active | 722 |
| 4.3.8.2.1.4 | Phase 3 Passive | 723 |
| 4.3.8.2.2 | Upstream Lanes | 723 |
| 4.3.8.2.2.1 | Phase 0 | 723 |
| 4.3.8.2.2.2 | Phase 1 Active | 723 |
| 4.3.8.2.2.3 | Phase 2 Active | 723 |
| 4.3.8.2.2.4 | Phase 2 Passive | 724 |
| 4.3.8.2.2.5 | Phase 3 | 724 |
| 4.3.8.2.3 | Force Timeout | 725 |
| 4.3.8.3 | Follower Loopback | 725 |
| 4.3.8.3.1 | Follower Loopback.Entry | 725 |
| 4.3.8.3.2 | Follower Loopback.Active | 726 |
| 4.3.8.3.3 | Follower Loopback.Exit | 726 |
| 4.3.9 | Retimer Latency | 726 |
| 4.3.9.1 | Measurement | 726 |
| 4.3.9.2 | Maximum Limit on Retimer Latency | 727 |
| 4.3.9.3 | Impacts on Upstream and Downstream Ports | 727 |
| 4.3.10 | SRIS | 727 |
| 4.3.11 | L1 PM Substates Support | 729 |
| 4.3.12 | Retimer Configuration Parameters | 730 |
| 4.3.12.1 | Global Parameters | 731 |
| 4.3.12.2 | Per Physical Pseudo Port Parameters | 731 |
| 4.3.13 | In Band Register Access | 732 |
| 5. | Power Management | 735 |
| 5.1 | Overview | 735 |
| 5.2 | Link State Power Management | 736 |
| 5.3 | PCI-PM Software Compatible Mechanisms | 740 |
| 5.3.1 | Device Power Management States (D-States) of a Function | 740 |
| 5.3.1.1 | D0 State | 741 |
| 5.3.1.2 | D1 State | 741 |
| 5.3.1.3 | D2 State | 741 |
| 5.3.1.4 | D3 State | 742 |
| 5.3.1.4.1 | D3 _{Hot} State | 742 |
| 5.3.1.4.2 | D3 _{Cold} State | 744 |
| 5.3.2 | PM Software Control of the Link Power Management State | 745 |
| 5.3.2.1 | Entry into the L1 State | 746 |
| 5.3.2.2 | Exit from L1 State | 748 |
| 5.3.2.3 | Entry into the L2/L3 Ready State | 749 |
| 5.3.3 | Power Management Event Mechanisms | 749 |
| 5.3.3.1 | Motivation | 749 |
| 5.3.3.2 | Link Wakeup | 750 |
| 5.3.3.2.1 | PME Synchronization | 751 |
| 5.3.3.3 | PM_PME Messages | 752 |

| | | |
|-----------|--|-----|
| 5.3.3.3.1 | PM_PME “Backpressure” Deadlock Avoidance | 753 |
| 5.3.3.4 | PME Rules | 753 |
| 5.3.3.5 | PM_PME Delivery State Machine | 754 |
| 5.4 | Native PCI Express Power Management Mechanisms | 755 |
| 5.4.1 | Active State Power Management (ASPM) | 755 |
| 5.4.1.1 | L0s ASPM State | 757 |
| 5.4.1.1.1 | Entry into the L0s State | 759 |
| 5.4.1.1.2 | Exit from the L0s State | 759 |
| 5.4.1.2 | ASPM L0p State | 760 |
| 5.4.1.3 | ASPM L1 State | 760 |
| 5.4.1.3.1 | ASPM Entry into the L1 State | 761 |
| 5.4.1.3.2 | Exit from the L1 State | 766 |
| 5.4.1.4 | ASPM Configuration | 768 |
| 5.4.1.4.1 | Software Flow for Enabling or Disabling ASPM | 771 |
| 5.5 | L1 PM Substates | 772 |
| 5.5.1 | Entry conditions for L1 PM Substates and L1.0 Requirements | 776 |
| 5.5.2 | L1.1 Requirements | 777 |
| 5.5.2.1 | Exit from L1.1 | 777 |
| 5.5.3 | L1.2 Requirements | 778 |
| 5.5.3.1 | L1.2.Entry | 779 |
| 5.5.3.2 | L1.2.Idle | 780 |
| 5.5.3.3 | L1.2.Exit | 780 |
| 5.5.3.3.1 | Exit from L1.2 | 781 |
| 5.5.4 | L1 PM Substates Configuration | 782 |
| 5.5.5 | L1 PM Substates Timing Parameters | 782 |
| 5.5.6 | Link Activation | 783 |
| 5.6 | Auxiliary Power Support | 784 |
| 5.7 | Power Management System Messages and DLLPs | 785 |
| 5.8 | PCI Function Power State Transitions | 786 |
| 5.9 | State Transition Recovery Time Requirements | 786 |
| 5.10 | SR-IOV Power Management | 786 |
| 5.10.1 | VF Device Power Management States | 787 |
| 5.10.2 | PF Device Power Management States | 787 |
| 5.11 | PCI Bridges and Power Management | 788 |
| 5.11.1 | Switches and PCI Express to PCI Bridges | 789 |
| 5.12 | Power Management Events | 789 |
| 6. | System Architecture | 791 |
| 6.1 | Interrupt and PME Support | 791 |
| 6.1.1 | Rationale for PCI Express Interrupt Model | 791 |
| 6.1.2 | PCI-compatible INTx Emulation | 792 |
| 6.1.3 | INTx Emulation Software Model | 792 |
| 6.1.4 | MSI and MSI-X Operation | 792 |
| 6.1.4.1 | MSI Configuration | 793 |
| 6.1.4.2 | MSI-X Configuration | 794 |
| 6.1.4.3 | Enabling Operation | 795 |
| 6.1.4.4 | Sending Messages | 796 |
| 6.1.4.5 | Per-vector Masking and Function Masking | 796 |
| 6.1.4.6 | Hardware/Software Synchronization | 797 |
| 6.1.4.7 | Message Transaction Reception and Ordering Requirements | 799 |
| 6.1.5 | PME Support | 799 |

| | | |
|-------------|---|---------------------------|
| 6.1.6 | Native PME Software Model..... | 799 |
| 6.1.7 | Legacy PME Software Model..... | 800 |
| 6.1.8 | Operating System Power Management Notification | 800 |
| 6.1.9 | PME Routing Between PCI Express and PCI Hierarchies | 800 |
| 6.2 | Error Signaling and Logging | 801 |
| 6.2.1 | Scope | 801 |
| 6.2.2 | Error Classification | 801 |
| 6.2.2.1 | Correctable Errors | 802 |
| 6.2.2.2 | Uncorrectable Errors..... | 803 |
| 6.2.2.2.1 | Fatal Errors | 803 |
| 6.2.2.2.2 | Non-Fatal Errors | 803 |
| 6.2.3 | Error Signaling..... | 803 |
| 6.2.3.1 | Completion Status | 804 |
| 6.2.3.2 | Error Messages | 804 |
| 6.2.3.2.1 | Uncorrectable Error Severity Programming (Advanced Error Reporting) | 805 |
| 6.2.3.2.2 | Masking Individual Errors | 805 |
| 6.2.3.2.3 | Error Pollution | 806 |
| 6.2.3.2.4 | Advisory Non-Fatal Error Cases | 806 |
| 6.2.3.2.4.1 | Completer Sending a Completion with UR/CA Status..... | 807 |
| 6.2.3.2.4.2 | Intermediate Receiver..... | 807 |
| 6.2.3.2.4.3 | Ultimate PCI Express Receiver of a Poisoned TLP or IDE TLP with PCRC Check Failed | 808 |
| 6.2.3.2.4.4 | Requester with Completion Timeout..... | 808 |
| 6.2.3.2.4.5 | Receiver of an Unexpected Completion..... | 809 |
| 6.2.3.2.5 | Requester Receiving a Completion with UR/CA Status | 809 |
| 6.2.3.3 | Error Forwarding (Data Poisoning)..... | 809 |
| 6.2.3.4 | Optional Error Checking | 809 |
| 6.2.4 | Error Logging..... | 810 |
| 6.2.4.1 | Root Complex Considerations (Advanced Error Reporting)..... | 811 |
| 6.2.4.1.1 | Error Source Identification | 811 |
| 6.2.4.1.2 | Interrupt Generation | 811 |
| 6.2.4.1.3 | ↑↑RC ECS Handling↑ | 812 |
| | | ECN: Base 6.3 RC-ECSΔ↔ |
| 6.2.4.2 | Multiple Error Handling (Advanced Error Reporting Extended Capability) | 812 |
| 6.2.4.2.1 | Multiple Error Handling in VFs | 814 |
| 6.2.4.3 | Advisory Non-Fatal Error Logging | 815 |
| 6.2.4.4 | End-End TLP Prefix Logging - Non-Flit Mode | 815 |
| 6.2.5 | Sequence of Device Error Signaling and Logging Operations..... | 816 |
| 6.2.6 | Error Message Controls | 818 |
| 6.2.7 | Error Listing and Rules..... | 819 |
| 6.2.7.1 | Conventional PCI Mapping | 824 |
| 6.2.8 | Virtual PCI Bridge Error Handling | 824 |
| 6.2.8.1 | Error Message Forwarding and PCI Mapping for Bridge - Rules..... | 825 |
| 6.2.9 | SR-IOV Baseline Error Handling | 825 |
| 6.2.10 | Internal Errors | 826 |
| 6.2.11 | Downstream Port Containment (DPC) | 827 |
| 6.2.11.1 | DPC Interrupts | 830 |
| 6.2.11.2 | DPC ERR_COR Signaling | 830 |
| 6.2.11.3 | Root Port Programmed I/O (RP PIO) Error Controls | 831 |
| 6.2.11.4 | Software Triggering of DPC..... | 834 |
| 6.2.11.5 | DL_Active ERR_COR Signaling..... | 835 |

| | | |
|-----------|---|-----|
| 6.3 | Virtual Channel Support | 836 |
| 6.3.1 | Introduction and Scope | 836 |
| 6.3.2 | TC/VC Mapping and Example Usage | 836 |
| 6.3.3 | VC Arbitration | 838 |
| 6.3.3.1 | Traffic Flow and Switch Arbitration Model..... | 839 |
| 6.3.3.2 | VC Arbitration - Arbitration Between VCs..... | 840 |
| 6.3.3.2.1 | Strict Priority Arbitration Model | 841 |
| 6.3.3.2.2 | Round Robin Arbitration Model..... | 841 |
| 6.3.3.3 | Port Arbitration - Arbitration Within VC..... | 842 |
| 6.3.3.4 | Multi-Function Devices and Function Arbitration..... | 842 |
| 6.3.4 | Isochronous Support | 845 |
| 6.3.4.1 | Rules for Software Configuration | 845 |
| 6.3.4.2 | Rules for Requesters | 846 |
| 6.3.4.3 | Rules for Completers..... | 846 |
| 6.3.4.4 | Rules for Switches and Root Complexes..... | 846 |
| 6.3.4.5 | Rules for Multi-Function Devices | 846 |
| 6.3.5 | SVC and VC/MFVC Capability Coexistence | 847 |
| 6.4 | Device Synchronization | 847 |
| 6.5 | Locked Transactions | 848 |
| 6.5.1 | Introduction | 848 |
| 6.5.2 | Initiation and Propagation of Locked Transactions - Rules | 849 |
| 6.5.3 | Switches and Lock - Rules..... | 850 |
| 6.5.4 | PCI Express/PCI Bridges and Lock - Rules | 850 |
| 6.5.5 | Root Complex and Lock - Rules | 850 |
| 6.5.6 | Legacy Endpoints | 850 |
| 6.5.7 | PCI Express Endpoints | 851 |
| 6.6 | PCI Express Reset - Rules | 851 |
| 6.6.1 | Conventional Reset | 851 |
| 6.6.2 | Function Level Reset (FLR)..... | 854 |
| 6.7 | PCI Express Native Hot-Plug | 858 |
| 6.7.1 | Elements of Hot-Plug | 858 |
| 6.7.1.1 | Indicators..... | 858 |
| 6.7.1.1.1 | Attention Indicator | 859 |
| 6.7.1.1.2 | Power Indicator | 860 |
| 6.7.1.2 | Manually-operated Retention Latch (MRL) | 860 |
| 6.7.1.3 | MRL Sensor | 860 |
| 6.7.1.4 | Electromechanical Interlock..... | 861 |
| 6.7.1.5 | Attention Button..... | 861 |
| 6.7.1.6 | Software User Interface..... | 862 |
| 6.7.1.7 | Slot Numbering | 862 |
| 6.7.1.8 | Power Controller | 862 |
| 6.7.2 | Registers Grouped by Hot-Plug Element Association..... | 863 |
| 6.7.2.1 | Attention Button Registers..... | 863 |
| 6.7.2.2 | Attention Indicator Registers..... | 863 |
| 6.7.2.3 | Power Indicator Registers | 863 |
| 6.7.2.4 | Power Controller Registers | 864 |
| 6.7.2.5 | Presence Detect Registers..... | 864 |
| 6.7.2.6 | MRL Sensor Registers | 864 |
| 6.7.2.7 | Electromechanical Interlock Registers..... | 864 |
| 6.7.2.8 | Command Completed Registers..... | 865 |
| 6.7.2.9 | Port Capabilities and Slot Information Registers | 865 |

Base 6.4 vs Base 6.3

| | | |
|---------------|---|------------|
| 6.7.2.10 | Hot-Plug Interrupt Control Register | 865 |
| 6.7.3 | PCI Express Hot-Plug Events | 865 |
| 6.7.3.1 | Slot Events | 866 |
| 6.7.3.2 | Command Completed Events | 866 |
| 6.7.3.3 | Data Link Layer State Changed Events | 866 |
| 6.7.3.4 | Software Notification of Hot-Plug Events | 867 |
| 6.7.4 | System Firmware Intermediary (SFI) Support | 868 |
| 6.7.4.1 | SFI ERR_COR Event Signaling | 868 |
| 6.7.4.2 | SFI Downstream Port Filtering (DPF) | 869 |
| 6.7.4.3 | SFI CAM | 870 |
| 6.7.4.4 | SFI Interactions with Readiness Notifications | 873 |
| 6.7.4.5 | SFI Suppression of Hot-Plug Surprise Functionality | 874 |
| 6.7.4.6 | ↑↑SFI Quarantine Mode↑↑ | 875 |
| 6.7.4.7 | ↑↑SFI Extended Capability In-Band Hiding↑↑ | 880 |
| 6.7.4.8 | ↑↑SFI Extended Capability In-Band Read-Only↑↑ | 881 |
| 6.7.5 | Firmware Support for Hot-Plug | 881 |
| 6.7.6 | Async Removal | 882 |
| 6.8 | Power Budgeting Mechanism | 883 |
| 6.8.1 | System Power Budgeting Process Recommendations | 884 |
| 6.8.2 | Device Power Considerations | 884 |
| 6.8.3 | Power Limit Mechanisms | 885 |
| 6.9 | Slot Power Limit Control | 886 |
| 6.10 | Root Complex Topology Discovery | 889 |
| 6.11 | Link Speed Management | 891 |
| 6.12 | Access Control Services (ACS) | 891 |
| 6.12.1 | ACS Component Capability Requirements | 892 |
| 6.12.1.1 | ACS Downstream Ports | 892 |
| 6.12.1.2 | ACS Functions in ↓↓SR-IOV Capable↓↓ ↑↑SR-IOV, SIOV, ↑↑ and Multi-Function Devices | 896 |
| 6.12.1.3 | Functions in Single-Function Devices | 897 |
| 6.12.2 | Interoperability | 897 |
| 6.12.3 | ACS Peer-to-Peer Control Interactions | 898 |
| 6.12.4 | ACS Enhanced Capability | 899 |
| 6.12.5 | ACS Violation Error Handling | 900 |
| 6.12.6 | ACS Redirection Impacts on Ordering Rules | 901 |
| 6.12.6.1 | Completions Passing Posted Requests | 901 |
| 6.12.6.2 | Requests Passing Posted Requests | 902 |
| 6.13 | Alternative Routing-ID Interpretation (ARI) | 903 |
| 6.14 | Multicast Operations | 906 |
| 6.14.1 | Multicast TLP Processing | 906 |
| 6.14.2 | Multicast Ordering | 908 |
| 6.14.3 | Multicast Capability Structure Field Updates | 909 |
| 6.14.4 | MC Blocked TLP Processing | 909 |
| 6.14.5 | MC_Overlay Mechanism | 909 |
| 6.15 | Atomic Operations (AtomicOps) | 912 |
| 6.15.1 | AtomicOp Use Models and Benefits | 913 |
| 6.15.2 | AtomicOp Transaction Protocol Summary | 914 |

ECN: Base 6.3
eSFI△↔

ECN: Base 6.3
eSFI△↔

ECN: Base 6.3
eSFI△↔

| | | |
|----------|---|-----|
| 6.15.3 | Root Complex Support for AtomicOps | 915 |
| 6.15.3.1 | Root Ports with AtomicOp Completer Capabilities | 915 |
| 6.15.3.2 | Root Ports with AtomicOp Routing Capability | 916 |
| 6.15.3.3 | RCs with AtomicOp Requester Capabilities | 916 |
| 6.15.4 | Switch Support for AtomicOps | 917 |
| 6.16 | Dynamic Power Allocation (DPA) Capability | 917 |
| 6.16.1 | DPA Capability with Multi-Function Devices | 918 |
| 6.17 | TLP Processing Hints (TPH) | 918 |
| 6.17.1 | Processing Hints | 918 |
| 6.17.2 | Steering Tags | 919 |
| 6.17.3 | ST Modes of Operation | 919 |
| 6.17.4 | TPH Capability | 920 |
| 6.18 | Latency Tolerance Reporting (LTR) Mechanism | 921 |
| 6.19 | Optimized Buffer Flush/Fill (OBFF) Mechanism | 926 |
| 6.20 | PASID | 929 |
| 6.20.1 | Managing PASID Usage | 929 |
| 6.20.2 | PASID Information Layout | 930 |
| 6.20.2.1 | PASID TLP Prefix - Non-Flit Mode | 930 |
| 6.20.2.2 | PASID field (Flit Mode and Non-Flit Mode) | 931 |
| 6.20.2.3 | Execute Requested | 932 |
| 6.20.2.4 | Privileged Mode Requested | 933 |
| 6.21 | Precision Time Measurement (PTM) Mechanism | 934 |
| 6.21.1 | Introduction | 934 |
| 6.21.2 | PTM Link Protocol | 935 |
| 6.21.3 | Configuration and Operational Requirements | 939 |
| 6.21.3.1 | PTM Requester Role | 940 |
| 6.21.3.2 | PTM Responder Role | 942 |
| 6.21.3.3 | PTM Time Source Role - Rules Specific to Switches | 943 |
| 6.22 | Readiness Notifications (RN) | 944 |
| 6.22.1 | Device Readiness Status (DRS) | 947 |
| 6.22.2 | Function Readiness Status (FRS) | 948 |
| 6.22.3 | FRS Queuing | 948 |
| 6.23 | Enhanced Allocation | 949 |
| 6.24 | Emergency Power Reduction State | 951 |
| 6.25 | Hierarchy ID Message | 954 |
| 6.26 | Flattening Portal Bridge (FPB) | 958 |
| 6.26.1 | Introduction | 958 |
| 6.26.2 | Hardware and Software Requirements | 962 |
| 6.27 | Vital Product Data (VPD) | 969 |
| 6.27.1 | VPD Format | 971 |
| 6.27.2 | VPD Definitions | 972 |
| 6.27.2.1 | VPD Large and Small Resource Data Tags | 972 |
| 6.27.2.2 | Read-Only Fields | 972 |
| 6.27.2.3 | Read/Write Fields | 974 |
| 6.27.2.4 | VPD Example | 974 |
| 6.28 | Native PCIe Enclosure Management | 975 |
| 6.29 | Conventional PCI Advanced Features Operation | 980 |
| 6.30 | Data Object Exchange (DOE) | 982 |
| 6.30.1 | Data Objects and Features | 985 |
| 6.30.1.1 | DOE Discovery Feature | 987 |
| 6.30.1.2 | DOE Async Message | 989 |

| | | |
|----------------|---|------|
| 6.30.2 | Operation..... | 990 |
| 6.30.3 | Interrupt Generation | 992 |
| 6.31 | Component Measurement and Authentication (CMA-SPDM) | 993 |
| 6.31.1 | Removed..... | 999 |
| 6.31.2 | Removed..... | 999 |
| 6.31.3 | CMA-SPDM Rules | 999 |
| 6.31.4 | Secured CMA-SPDM..... | 1001 |
| 6.32 | Deferrable Memory Write..... | 1002 |
| 6.33 | Integrity & Data Encryption (IDE)..... | 1007 |
| 6.33.1 | IDE Stream and TEE State Machines..... | 1011 |
| 6.33.2 | IDE Stream Establishment | 1013 |
| 6.33.3 | IDE Key Management (IDE_KM) | 1014 |
| 6.33.4 | IDE TLPs | 1024 |
| 6.33.5 | IDE TLP Sub-Streams | 1042 |
| 6.33.6 | IDE TLP Aggregation..... | 1047 |
| 6.33.7 | Flow-Through Selective IDE Streams | 1048 |
| 6.33.8 | Other IDE Rules..... | 1050 |
| 6.34 | ↓↑Unordered I/O↓↑Unordered I/O↑ (UIO) | 1052 |
| 6.34.1 | UIO Rules | 1053 |
| 6.35 | MMIO Register Blocks..... | 1053 |
| 6.35.1 | MMIO Capabilities Register Block (MCAP) | 1054 |
| 6.35.1.1 | MCAP Array Register (Offset 00h)..... | 1055 |
| 6.35.1.2 | MCAP Header Register Block (Offset Varies) | 1057 |
| 6.35.1.3 | MMIO Mailbox Capability (MMB) (Offset: Varies) | 1059 |
| 6.35.1.3.1 | MMB Operation..... | 1060 |
| 6.35.1.3.2 | MMB Registers | 1061 |
| 6.35.1.3.2.1 | MMB Capabilities Register (Offset 00h) | 1061 |
| 6.35.1.3.2.2 | MMB Control Register (Offset 04h) | 1063 |
| 6.35.1.3.2.3 | MMB Command Register (Offset 08h) | 1064 |
| 6.35.1.3.2.4 | MMB Status Register (Offset 10h) | 1065 |
| 6.35.1.3.2.4.1 | MMB Command Return Codes | 1066 |
| 6.35.1.3.2.5 | MMB Payload Registers (Offset 20h)..... | 1066 |
| 6.35.1.4 | Management Message Passthrough (MMPT) Capability (Offset: Varies) | 1067 |
| 6.35.1.4.1 | MMPT Registers | 1067 |
| 6.35.1.4.1.1 | MMPT Capabilities Register (Offset 00h) | 1067 |
| 6.35.1.4.1.2 | MMPT Control Register (Offset 04h) | 1068 |
| 6.35.1.4.1.3 | MMPT Receive Message Notification Register (Offset 08h)..... | 1069 |
| 6.35.2 | MMIO Designated Vendor-Specific Register Block (MDVS) | 1069 |
| 6.35.2.1 | MDVS Register Block Header Register 1 (Offset 00h) | 1070 |
| 6.35.3 | MDVS Register Block Header Register 2 (Offset 04h) | 1071 |
| 6.35.4 | MDVS Register Block Header Register 3 (Offset 08h) | 1071 |
| 6.36 | MMB Command Interface | 1072 |
| 6.36.1 | Management Message Passthrough (MMPT) | 1072 |
| 6.36.1.1 | MMPT Send Message (Opcode 0100h)..... | 1073 |
| 6.36.1.1.1 | MMPT Send Message Operation | 1074 |
| 6.36.1.2 | MMPT Receive Message (Opcode 0101h) | 1075 |
| 6.36.1.2.1 | MMPT Receive Message Operation | 1076 |
| 6.37 | Debug Over Link | 1076 |
| 6.37.1 | NOP Flit..... | 1076 |
| 7. | Software Initialization and Configuration | 1079 |

| | | |
|------------|---|------|
| 7.1 | Configuration Topology | 1079 |
| 7.2 | PCI Express Configuration Mechanisms | 1080 |
| 7.2.1 | PCI-compatible Configuration Mechanism | 1081 |
| 7.2.2 | PCI Express Enhanced Configuration Access Mechanism (ECAM) | 1082 |
| 7.2.2.1 | Host Bridge Requirements | 1085 |
| 7.2.2.2 | PCI Express Device Requirements | 1085 |
| 7.2.3 | Root Complex Register Block (RCRB) | 1086 |
| 7.3 | Configuration Transaction Rules | 1086 |
| 7.3.1 | Device Number | 1086 |
| 7.3.2 | Configuration Transaction Addressing | 1087 |
| 7.3.3 | Configuration Request Routing Rules | 1087 |
| 7.3.4 | PCI Special Cycles | 1089 |
| 7.4 | Configuration Register Types | 1089 |
| 7.5 | PCI and PCIe Capabilities Required by the Base Spec for all Ports | 1091 |
| 7.5.1 | PCI-Compatible Configuration Registers | 1091 |
| 7.5.1.1 | Type 0/1 Common Configuration Space | 1091 |
| 7.5.1.1.1 | Vendor ID Register (Offset 00h) | 1092 |
| 7.5.1.1.2 | Device ID Register (Offset 02h) | 1093 |
| 7.5.1.1.3 | Command Register (Offset 04h) | 1093 |
| 7.5.1.1.4 | Status Register (Offset 06h) | 1096 |
| 7.5.1.1.5 | Revision ID Register (Offset 08h) | 1098 |
| 7.5.1.1.6 | Class Code Register (Offset 09h) | 1098 |
| 7.5.1.1.7 | Cache Line Size Register (Offset 0Ch) | 1099 |
| 7.5.1.1.8 | Latency Timer Register (Offset 0Dh) | 1099 |
| 7.5.1.1.9 | Header Type Register (Offset 0Eh) | 1099 |
| 7.5.1.1.10 | BIST Register (Offset 0Fh) | 1100 |
| 7.5.1.1.11 | Capabilities Pointer (Offset 34h) | 1101 |
| 7.5.1.1.12 | Interrupt Line Register (Offset 3Ch) | 1101 |
| 7.5.1.1.13 | Interrupt Pin Register (Offset 3Dh) | 1102 |
| 7.5.1.1.14 | Error Registers | 1102 |
| 7.5.1.2 | Type 0 Configuration Space Header | 1103 |
| 7.5.1.2.1 | Base Address Registers (Offset 10h - 24h) | 1103 |
| 7.5.1.2.2 | Cardbus CIS Pointer Register (Offset 28h) | 1107 |
| 7.5.1.2.3 | Subsystem Vendor ID Register / Subsystem ID Register (Offset 2Ch/2Eh) | 1107 |
| 7.5.1.2.4 | Expansion ROM Base Address Register (Offset 30h) | 1108 |
| 7.5.1.2.5 | Min_Gnt Register / Max_Lat Register (Offset 3Eh/3Fh) | 1111 |
| 7.5.1.3 | Type 1 Configuration Space Header | 1111 |
| 7.5.1.3.1 | Type 1 Base Address Registers (Offset 10h-14h) | 1113 |
| 7.5.1.3.2 | Primary Bus Number Register (Offset 18h) | 1113 |
| 7.5.1.3.3 | Secondary Bus Number Register (Offset 19h) | 1113 |
| 7.5.1.3.4 | Subordinate Bus Number Register (Offset 1Ah) | 1113 |
| 7.5.1.3.5 | Secondary Latency Timer (Offset 1Bh) | 1113 |
| 7.5.1.3.6 | I/O Base / I/O Limit Registers (Offset 1Ch/1Dh) | 1114 |
| 7.5.1.3.7 | Secondary Status Register (Offset 1Eh) | 1114 |
| 7.5.1.3.8 | Memory Base Register / Memory Limit Register (Offset 20h/22h) | 1116 |
| 7.5.1.3.9 | 64-bit Memory Base / 64-bit Memory Limit Registers (Offset 24h/26h) and 64-bit Base Upper 32 Bits / 64-bit Limit Upper 32 Bits Registers (Offset 28h/2Ch) | 1116 |
| 7.5.1.3.10 | 64-bit Base Upper 32 Bits / 64-bit Limit Upper 32 Bits Registers (Offset 28h/2Ch) | 1117 |
| 7.5.1.3.11 | I/O Base Upper 16 Bits / I/O Limit Upper 16 Bits Registers (Offset 30h/32h) | 1117 |
| 7.5.1.3.12 | Expansion ROM Base Address Register (Offset 38h) | 1117 |
| 7.5.1.3.13 | Bridge Control Register (Offset 3Eh) | 1118 |

| | | |
|--------------|---|-------------|
| 7.5.2 | PCI Power Management Capability Structure | 1120 |
| 7.5.2.1 | Power Management Capabilities Register (Offset 00h) | 1121 |
| 7.5.2.2 | Power Management Control/Status Register (Offset 04h) | 1123 |
| 7.5.2.3 | Power Management Data Register (Offset 07h) | 1124 |
| 7.5.3 | PCI Express Capability Structure | 1126 |
| 7.5.3.1 | PCI Express Capability List Register (Offset 00h) | 1127 |
| 7.5.3.2 | PCI Express Capabilities Register (Offset 02h) | 1128 |
| 7.5.3.3 | Device Capabilities Register (Offset 04h) | 1130 |
| 7.5.3.4 | Device Control Register (Offset 08h)..... | 1134 |
| 7.5.3.5 | Device Status Register (Offset 0Ah) | 1141 |
| 7.5.3.6 | Link Capabilities Register (Offset 0Ch) | 1143 |
| 7.5.3.7 | Link Control Register (Offset 10h)..... | 1146 |
| 7.5.3.8 | Link Status Register (Offset 12h)..... | 1153 |
| 7.5.3.9 | Slot Capabilities Register (Offset 14h)..... | 1156 |
| 7.5.3.10 | Slot Control Register (Offset 18h) | 1158 |
| 7.5.3.11 | Slot Status Register (Offset 1Ah) | 1161 |
| 7.5.3.12 | Root Control Register (Offset 1Ch) | 1163 |
| 7.5.3.13 | Root Capabilities Register (Offset 1Eh) | 1165 |
| 7.5.3.14 | Root Status Register (Offset 20h) | 1165 |
| 7.5.3.15 | Device Capabilities 2 Register (Offset 24h)..... | 1167 |
| 7.5.3.16 | Device Control 2 Register (Offset 28h)..... | 1172 |
| 7.5.3.17 | Device Status 2 Register (Offset 2Ah)..... | 1175 |
| 7.5.3.18 | Link Capabilities 2 Register (Offset 2Ch) | 1176 |
| 7.5.3.19 | Link Control 2 Register (Offset 30h)..... | 1179 |
| 7.5.3.20 | Link Status 2 Register (Offset 32h) | 1182 |
| 7.5.3.21 | Slot Capabilities 2 Register (Offset 34h) | 1186 |
| 7.5.3.22 | Slot Control 2 Register (Offset 38h) | 1187 |
| 7.5.3.23 | Slot Status 2 Register (Offset 3Ah) | 1187 |
| 7.6 | PCI Express Extended Capabilities | 1187 |
| 7.6.1 | Extended Capabilities in Configuration Space..... | 1188 |
| 7.6.2 | Extended Capabilities in the Root Complex Register Block | 1188 |
| 7.6.3 | PCI Express Extended Capability Header | 1188 |
| 7.7 | PCI and PCIe Capabilities Required by the Base Spec in Some Situations | 1189 |
| 7.7.1 | MSI Capability Structures..... | 1189 |
| 7.7.1.1 | MSI Capability Header (Offset 00h) | 1191 |
| 7.7.1.2 | Message Control Register for MSI (Offset 02h) | 1191 |
| 7.7.1.3 | Message Address Register for MSI (Offset 04h) | 1193 |
| 7.7.1.4 | Message Upper Address Register for MSI (Offset 08h) | 1194 |
| 7.7.1.5 | Message Data Register for MSI (Offset 08h or 0Ch) | 1194 |
| 7.7.1.6 | Extended Message Data Register for MSI (Optional)..... | 1195 |
| 7.7.1.7 | Mask Bits Register for MSI (Offset 0Ch or 10h) | 1195 |
| 7.7.1.8 | Pending Bits Register for MSI (Offset 10h or 14h) | 1196 |
| 7.7.2 | MSI-X Capability and Table Structure | 1196 |
| 7.7.2.1 | MSI-X Capability Header (Offset 00h) | 1200 |
| 7.7.2.2 | Message Control Register for MSI-X (Offset 02h) | 1201 |
| 7.7.2.3 | Table Offset/Table BIR Register for MSI-X (Offset 04h) | 1202 |
| 7.7.2.4 | PBA Offset/PBA BIR Register for MSI-X (Offset 08h) | 1202 |
| 7.7.2.5 | Message Address Register for MSI-X Table Entries..... | 1203 |
| 7.7.2.6 | Message Upper Address Register for MSI-X Table Entries | 1204 |
| 7.7.2.7 | Message Data Register for MSI-X Table Entries | 1204 |
| 7.7.2.8 | Vector Control Register for MSI-X Table Entries | 1204 |

| | | |
|--------------|--|-------------|
| 7.7.2.9 | Pending Bits Register for MSI-X PBA Entries | 1205 |
| 7.7.3 | Secondary PCI Express Extended Capability..... | 1206 |
| 7.7.3.1 | Secondary PCI Express Extended Capability Header (Offset 00h) | 1208 |
| 7.7.3.2 | Link Control 3 Register (Offset 04h)..... | 1208 |
| 7.7.3.3 | Lane Error Status Register (Offset 08h) | 1209 |
| 7.7.3.4 | Lane Equalization Control Register (Offset 0Ch)..... | 1210 |
| 7.7.4 | Data Link Feature Extended Capability | 1212 |
| 7.7.4.1 | Data Link Feature Extended Capability Header (Offset 00h) | 1213 |
| 7.7.4.2 | Data Link Feature Capabilities Register (Offset 04h) | 1214 |
| 7.7.4.3 | Data Link Feature Status Register (Offset 08h)..... | 1215 |
| 7.7.5 | Physical Layer 16.0 GT/s Extended Capability..... | 1216 |
| 7.7.5.1 | Physical Layer 16.0 GT/s Extended Capability Header (Offset 00h) | 1218 |
| 7.7.5.2 | 16.0 GT/s Capabilities Register (Offset 04h) | 1218 |
| 7.7.5.3 | 16.0 GT/s Control Register (Offset 08h)..... | 1219 |
| 7.7.5.4 | 16.0 GT/s Status Register (Offset 0Ch) | 1219 |
| 7.7.5.5 | 16.0 GT/s Local Data Parity Mismatch Status Register (Offset 10h) | 1220 |
| 7.7.5.6 | 16.0 GT/s First Retimer Data Parity Mismatch Status Register (Offset 14h) | 1221 |
| 7.7.5.7 | 16.0 GT/s Second Retimer Data Parity Mismatch Status Register (Offset 18h) | 1221 |
| 7.7.5.8 | Physical Layer 16.0 GT/s Reserved (Offset 1Ch) | 1222 |
| 7.7.5.9 | 16.0 GT/s Lane Equalization Control Register (Offsets 20h to 3Ch)..... | 1222 |
| 7.7.6 | Physical Layer 32.0 GT/s Extended Capability..... | 1223 |
| 7.7.6.1 | Physical Layer 32.0 GT/s Extended Capability Header (Offset 00h) | 1225 |
| 7.7.6.2 | 32.0 GT/s Capabilities Register (Offset 04h) | 1225 |
| 7.7.6.3 | 32.0 GT/s Control Register (Offset 08h)..... | 1226 |
| 7.7.6.4 | 32.0 GT/s Status Register (Offset 0Ch) | 1227 |
| 7.7.6.5 | Received Modified TS Data 1 Register (Offset 10h) | 1228 |
| 7.7.6.6 | Received Modified TS Data 2 Register (Offset 14h) | 1229 |
| 7.7.6.7 | Transmitted Modified TS Data 1 Register (Offset 18h) | 1230 |
| 7.7.6.8 | Transmitted Modified TS Data 2 Register (Offset 1Ch) | 1231 |
| 7.7.6.9 | 32.0 GT/s Lane Equalization Control Register (Offset 20h) | 1232 |
| 7.7.7 | Physical Layer 64.0 GT/s Extended Capability..... | 1234 |
| 7.7.7.1 | Physical Layer 64.0 GT/s Extended Capability Header (Offset 00h) | 1235 |
| 7.7.7.2 | 64.0 GT/s Capabilities Register (Offset 04h) | 1235 |
| 7.7.7.3 | 64.0 GT/s Control Register (Offset 08h)..... | 1236 |
| 7.7.7.4 | 64.0 GT/s Status Register (Offset 0Ch) | 1236 |
| 7.7.7.5 | 64.0 GT/s Lane Equalization Control Register (Offset 10h) | 1237 |
| 7.7.8 | Flit Logging Extended Capability..... | 1239 |
| 7.7.8.1 | Flit Logging Extended Capability Header (Offset 00h)..... | 1240 |
| 7.7.8.2 | Flit Error Log 1 Register (Offset 04h)..... | 1240 |
| 7.7.8.3 | Flit Error Log 2 Register (Offset 08h)..... | 1243 |
| 7.7.8.4 | Flit Error Counter Control Register (Offset 0Ch) | 1243 |
| 7.7.8.5 | Flit Error Counter Status Register (Offset 0Eh)..... | 1245 |
| 7.7.8.6 | FBER Measurement Control Register (Offset 10h) | 1246 |
| 7.7.8.7 | FBER Measurement Status 1 Register (Offset 14h) | 1246 |
| 7.7.8.8 | FBER Measurement Status 2 Register (Offset 18h) | 1247 |
| 7.7.8.9 | FBER Measurement Status 3 Register (Offset 1Ch) | 1248 |
| 7.7.8.10 | FBER Measurement Status 4 Register (Offset 20h) | 1248 |
| 7.7.8.11 | FBER Measurement Status 5 Register (Offset 24h) | 1249 |
| 7.7.8.12 | FBER Measurement Status 6 Register (Offset 28h) | 1249 |
| 7.7.8.13 | FBER Measurement Status 7 Register (Offset 2Ch) | 1249 |
| 7.7.8.14 | FBER Measurement Status 8 Register (Offset 30h) | 1250 |

| | | |
|---------------|--|----------------------------|
| 7.7.8.15 | FBER Measurement Status 9 Register (Offset 34h) | 1250 |
| 7.7.8.16 | FBER Measurement Status 10 Register (Offset 38h) | 1251 |
| 7.7.9 | Device 3 Extended Capability Structure | 1251 |
| 7.7.9.1 | Device 3 Extended Capability Header (Offset 00h) | 1251 |
| 7.7.9.2 | Device Capabilities 3 Register (Offset 04h)..... | 1252 |
| 7.7.9.3 | Device Control 3 Register (Offset 08h)..... | 1255 |
| 7.7.9.4 | Device Status 3 Register (Offset 0Ch)..... | 1257 |
| 7.7.10 | Lane Margining at the Receiver Extended Capability | 1258 |
| 7.7.10.1 | Lane Margining at the Receiver Extended Capability Header (Offset 00h) | 1261 |
| 7.7.10.2 | Margining Port Capabilities Register (Offset 04h)..... | 1261 |
| 7.7.10.3 | Margining Port Status Register (Offset 06h) | 1262 |
| 7.7.10.4 | Margining Lane Control Register (Offset 08h) | 1262 |
| 7.7.10.5 | Margining Lane Status Register (Offset 0Ah) | 1263 |
| 7.7.11 | ACS Extended Capability..... | 1264 |
| 7.7.11.1 | ACS Extended Capability Header (Offset 00h)..... | 1265 |
| 7.7.11.2 | ACS Capability Register (Offset 04h)..... | 1266 |
| 7.7.11.3 | ACS Control Register (Offset 06h) | 1267 |
| 7.7.11.4 | Egress Control Vector Register (Offset 08h) | 1269 |
| 7.8 | Common PCI and PCIe Capabilities | 1271 |
| 7.8.1 | Power Budgeting Extended Capability..... | 1271 |
| 7.8.1.1 | Power Budgeting Extended Capability Header (Offset 00h) | 1271 |
| 7.8.1.2 | Power Budgeting Data Select Register (Offset 04h)..... | 1272 |
| 7.8.1.3 | Power Budgeting Control Register (Offset 06h) | 1272 |
| 7.8.1.4 | Power Budgeting Data Register (Offset 08h) | 1274 |
| 7.8.1.5 | Power Budgeting Capability Register (Offset 0Ch) | 1279 |
| 7.8.1.6 | Power Budgeting Sense Detect Register (Offset 0Dh) | 1280 |
| 7.8.2 | Latency Tolerance Reporting (LTR) Extended Capability..... | 1283 |
| 7.8.2.1 | LTR Extended Capability Header (Offset 00h) | 1284 |
| 7.8.2.2 | Max Snoop Latency Register (Offset 04h)..... | 1285 |
| 7.8.2.3 | Max No-Snoop Latency Register (Offset 06h)..... | 1285 |
| 7.8.2.4 | ↑↓LTR Capabilities Register (Offset 08h)↑ | 1286 |
| | | ECN: Base 6.3 LTR-MFD△◀ |
| 7.8.3 | L1 PM Substates Extended Capability | 1286 |
| 7.8.3.1 | L1 PM Substates Extended Capability Header (Offset 00h) | 1287 |
| 7.8.3.2 | L1 PM Substates Capabilities Register (Offset 04h) | 1288 |
| 7.8.3.3 | L1 PM Substates Control 1 Register (Offset 08h) | 1289 |
| 7.8.3.4 | L1 PM Substates Control 2 Register (Offset 0Ch) | 1291 |
| 7.8.3.5 | L1 PM Substates Status Register (Offset 10h)..... | 1292 |
| 7.8.4 | ↑↓Advanced Error Reporting Extended Capability↑ ↑↓Advanced Error Reporting Extended Capability↑ ↑↓(AER)↑ | 1292 |
| 7.8.4.1 | Advanced Error Reporting Extended Capability Header (Offset 00h) | 1295 |
| 7.8.4.2 | Uncorrectable Error Status Register (Offset 04h)..... | 1295 |
| 7.8.4.3 | Uncorrectable Error Mask Register (Offset 08h)..... | 1298 |
| 7.8.4.4 | Uncorrectable Error Severity Register (Offset 0Ch) | 1300 |
| 7.8.4.5 | Correctable Error Status Register (Offset 10h) | 1303 |
| 7.8.4.6 | Correctable Error Mask Register (Offset 14h) | 1304 |
| 7.8.4.7 | Advanced Error Capabilities and Control Register (Offset 18h) | 1305 |
| 7.8.4.8 | Header Log Register (Offset 1Ch)..... | 1307 |
| 7.8.4.9 | Root Error Command Register (Offset 2Ch) | 1308 |
| 7.8.4.10 | Root Error Status Register (Offset 30h)..... | 1311 |
| 7.8.4.11 | Error Source Identification Register (Offset 34h) | 1313 |

| | | |
|---------------|--|-------------|
| 7.8.4.12 | TLP Prefix Log Register (Offset 38h) | 1314 |
| 7.8.5 | Enhanced Allocation Capability Structure (EA) | 1315 |
| 7.8.5.1 | Enhanced Allocation Capability First DW (Offset 00h)..... | 1315 |
| 7.8.5.2 | Enhanced Allocation Capability Second DW (Offset 04h) [Type 1 Functions Only] | 1315 |
| 7.8.5.3 | Enhanced Allocation Per-Entry Format (Offset 04h or 08h) | 1316 |
| 7.8.6 | Resizable BAR Extended Capability | 1321 |
| 7.8.6.1 | Resizable BAR Extended Capability Header (Offset 00h)..... | 1323 |
| 7.8.6.2 | Resizable BAR Capability Register | 1323 |
| 7.8.6.3 | Resizable BAR Control Register | 1326 |
| 7.8.7 | VF Resizable BAR Extended Capability | 1328 |
| 7.8.7.1 | VF Resizable BAR Extended Capability Header (Offset 00h)..... | 1330 |
| 7.8.7.2 | VF Resizable BAR Capability Register (Offset 04h) | 1330 |
| 7.8.7.3 | VF Resizable BAR Control Register (Offset 08h) | 1330 |
| 7.8.8 | ARI Extended Capability | 1332 |
| 7.8.8.1 | ARI Extended Capability Header (Offset 00h)..... | 1332 |
| 7.8.8.2 | ARI Capability Register (Offset 04h) | 1333 |
| 7.8.8.3 | ARI Control Register (Offset 06h) | 1334 |
| 7.8.9 | PASID Extended Capability Structure | 1334 |
| 7.8.9.1 | PASID Extended Capability Header (Offset 00h) | 1335 |
| 7.8.9.2 | PASID Capability Register (Offset 04h) | 1336 |
| 7.8.9.3 | PASID Control Register (Offset 06h) | 1336 |
| 7.8.10 | FRS Queueing Extended Capability | 1338 |
| 7.8.10.1 | FRS Queueing Extended Capability Header (Offset 00h)..... | 1338 |
| 7.8.10.2 | FRS Queueing Capability Register (Offset 04h) | 1339 |
| 7.8.10.3 | FRS Queueing Status Register (Offset 08h) | 1340 |
| 7.8.10.4 | FRS Queueing Control Register (Offset 0Ah) | 1340 |
| 7.8.10.5 | FRS Message Queue Register (Offset 0Ch) | 1341 |
| 7.8.11 | Flattening Portal Bridge (FPB) Capability | 1341 |
| 7.8.11.1 | FPB Capability Header (Offset 00h) | 1342 |
| 7.8.11.2 | FPB Capabilities Register (Offset 04h) | 1343 |
| 7.8.11.3 | FPB RID Vector Control 1 Register (Offset 08h) | 1345 |
| 7.8.11.4 | FPB RID Vector Control 2 Register (Offset 0Ch) | 1346 |
| 7.8.11.5 | FPB MEM Low Vector Control Register (Offset 10h) | 1347 |
| 7.8.11.6 | FPB MEM High Vector Control 1 Register (Offset 14h) | 1348 |
| 7.8.11.7 | FPB MEM High Vector Control 2 Register (Offset 18h) | 1350 |
| 7.8.11.8 | FPB Vector Access Control Register (Offset 1Ch) | 1350 |
| 7.8.11.9 | FPB Vector Access Data Register (Offset 20h) | 1352 |
| 7.8.12 | Flit Performance Measurement Extended Capability | 1352 |
| 7.8.12.1 | Flit Performance Measurement Extended Capability Header (Offset 00h)..... | 1353 |
| 7.8.12.2 | Flit Performance Measurement Capability Register (Offset 04h) | 1354 |
| 7.8.12.3 | Flit Performance Measurement Control Register (Offset 08h) | 1354 |
| 7.8.12.4 | Flit Performance Measurement Status Register (Offset 0Ch) | 1357 |
| 7.8.12.5 | LTSSM Performance Measurement Status Register (Offsets 10h to 20h) | 1358 |
| 7.8.13 | Flit Error Injection Extended Capability | 1359 |
| 7.8.13.1 | Flit Error Injection Extended Capability Header (Offset 00h) | 1360 |
| 7.8.13.2 | Flit Error Injection Capability Register (Offset 04h) | 1361 |
| 7.8.13.3 | Flit Error Injection Control 1 Register (Offset 08h) | 1361 |
| 7.8.13.4 | Flit Error Injection Control 2 Register (Offset 0Ch) | 1363 |
| 7.8.13.5 | Flit Error Injection Status Register (Offset 10h) | 1364 |
| 7.8.13.6 | Ordered Set Error Injection Control 1 Register (Offset 14h) | 1365 |

| | | |
|---------------|--|-------------|
| 7.8.13.7 | Ordered Set Error Injection Control 2 Register (Offset 18h) | 1366 |
| 7.8.13.8 | Ordered Set Error Tx Injection Status Register (Offset 1Ch) | 1367 |
| 7.8.13.9 | Ordered Set Error Rx Injection Status Register (Offset 20h) | 1368 |
| 7.8.14 | NOP Flit Extended Capability..... | 1369 |
| 7.8.14.1 | NOP Flit Extended Capability Header..... | 1370 |
| 7.8.14.2 | NOP Flit Capabilities Register | 1371 |
| 7.8.14.3 | NOP Flit Control 1 Register | 1371 |
| 7.8.14.4 | NOP Flit Control 2 Register | 1373 |
| 7.8.14.5 | NOP Flit Status Register | 1374 |
| 7.9 | Additional PCI and PCIe Capabilities..... | 1375 |
| 7.9.1 | Virtual Channel Extended Capability..... | 1375 |
| 7.9.1.1 | Virtual Channel Extended Capability Header (Offset 00h) | 1376 |
| 7.9.1.2 | Port VC Capability Register 1 (Offset 04h) | 1377 |
| 7.9.1.3 | Port VC Capability Register 2 (Offset 08h) | 1378 |
| 7.9.1.4 | Port VC Control Register (Offset 0Ch) | 1378 |
| 7.9.1.5 | Port VC Status Register (Offset 0Eh) | 1379 |
| 7.9.1.6 | VC Resource Capability Register | 1380 |
| 7.9.1.7 | VC Resource Control Register | 1381 |
| 7.9.1.8 | VC Resource Status Register | 1384 |
| 7.9.1.9 | VC Arbitration Table | 1385 |
| 7.9.1.10 | Port Arbitration Table..... | 1385 |
| 7.9.2 | Multi-Function Virtual Channel Extended Capability | 1387 |
| 7.9.2.1 | MFVC Extended Capability Header (Offset 00h) | 1387 |
| 7.9.2.2 | MFVC Port VC Capability Register 1 (Offset 04h) | 1388 |
| 7.9.2.3 | MFVC Port VC Capability Register 2 (Offset 08h) | 1389 |
| 7.9.2.4 | MFVC Port VC Control Register (Offset 0Ch) | 1390 |
| 7.9.2.5 | MFVC Port VC Status Register (Offset 0Eh) | 1391 |
| 7.9.2.6 | MFVC VC Resource Capability Register | 1391 |
| 7.9.2.7 | MFVC VC Resource Control Register | 1392 |
| 7.9.2.8 | MFVC VC Resource Status Register | 1395 |
| 7.9.2.9 | MFVC VC Arbitration Table | 1395 |
| 7.9.2.10 | Function Arbitration Table | 1396 |
| 7.9.3 | Device Serial Number Extended Capability | 1397 |
| 7.9.3.1 | Device Serial Number Extended Capability Header (Offset 00h) | 1398 |
| 7.9.3.2 | Serial Number Register (Offset 04h) | 1399 |
| 7.9.4 | Vendor-Specific Capability..... | 1399 |
| 7.9.5 | Vendor-Specific Extended Capability | 1400 |
| 7.9.5.1 | Vendor-Specific Extended Capability Header (Offset 00h) | 1401 |
| 7.9.5.2 | Vendor-Specific Header (Offset 04h) | 1402 |
| 7.9.6 | Designated Vendor-Specific Extended Capability (DVSEC) | 1402 |
| 7.9.6.1 | Designated Vendor-Specific Extended Capability Header (Offset 00h) | 1403 |
| 7.9.6.2 | Designated Vendor-Specific Header 1 (Offset 04h)..... | 1404 |
| 7.9.6.3 | Designated Vendor-Specific Header 2 (Offset 08h)..... | 1404 |
| 7.9.7 | RCRB Header Extended Capability | 1405 |
| 7.9.7.1 | RCRB Header Extended Capability Header (Offset 00h) | 1405 |
| 7.9.7.2 | RCRB Vendor ID and Device ID register (Offset 04h) | 1406 |
| 7.9.7.3 | RCRB Capabilities register (Offset 08h) | 1406 |
| 7.9.7.4 | RCRB Control register (Offset 0Ch) | 1407 |
| 7.9.8 | Root Complex Link Declaration Extended Capability | 1407 |
| 7.9.8.1 | Root Complex Link Declaration Extended Capability Header (Offset 00h)..... | 1409 |
| 7.9.8.2 | Element Self Description Register (Offset 04h) | 1410 |

| | | |
|-------------|--|------|
| 7.9.8.3 | Link Entries | 1411 |
| 7.9.8.3.1 | Link Description Register | 1411 |
| 7.9.8.3.2 | Link Address | 1412 |
| 7.9.8.3.2.1 | Link Address for Link Type 0 | 1412 |
| 7.9.8.3.2.2 | Link Address for Link Type 1 | 1412 |
| 7.9.9 | Root Complex Internal Link Control Extended Capability | 1413 |
| 7.9.9.1 | Root Complex Internal Link Control Extended Capability Header (Offset 00h) | 1414 |
| 7.9.9.2 | Root Complex Link Capabilities Register (Offset 04h) | 1414 |
| 7.9.9.3 | Root Complex Link Control Register (Offset 08h) | 1417 |
| 7.9.9.4 | Root Complex Link Status Register (Offset 0Ah) | 1418 |
| 7.9.10 | Root Complex Event Collector Endpoint Association Extended Capability | 1419 |
| 7.9.10.1 | Root Complex Event Collector Endpoint Association Extended Capability Header (Offset 00h) .. | 1420 |
| 7.9.10.2 | Association Bitmap for RCiEPs (Offset 04h) | 1421 |
| 7.9.10.3 | RCEC Associated Bus Numbers Register (Offset 08h) | 1421 |
| 7.9.11 | Multicast Extended Capability | 1422 |
| 7.9.11.1 | Multicast Extended Capability Header (Offset 00h) | 1423 |
| 7.9.11.2 | Multicast Capability Register (Offset 04h) | 1424 |
| 7.9.11.3 | Multicast Control Register (Offset 06h) | 1425 |
| 7.9.11.4 | MC_Base_Address Register (Offset 08h) | 1425 |
| 7.9.11.5 | MC_Receive Register (Offset 10h) | 1426 |
| 7.9.11.6 | MC_Block_All Register (Offset 18h) | 1426 |
| 7.9.11.7 | MC_Block_Untranslated Register (Offset 20h) | 1427 |
| 7.9.11.8 | MC_Overlay_BAR Register (Offset 28h) | 1427 |
| 7.9.12 | Dynamic Power Allocation Extended Capability (DPA Capability) | 1428 |
| 7.9.12.1 | DPA Extended Capability Header (Offset 00h) | 1429 |
| 7.9.12.2 | DPA Capability Register (Offset 04h) | 1429 |
| 7.9.12.3 | DPA Latency Indicator Register (Offset 08h) | 1430 |
| 7.9.12.4 | DPA Status Register (Offset 0Ch) | 1431 |
| 7.9.12.5 | DPA Control Register (Offset 0Eh) | 1431 |
| 7.9.12.6 | DPA Power Allocation Array | 1432 |
| 7.9.13 | TPH Requester Extended Capability | 1432 |
| 7.9.13.1 | TPH Requester Extended Capability Header (Offset 00h) | 1433 |
| 7.9.13.2 | TPH Requester Capability Register (Offset 04h) | 1433 |
| 7.9.13.3 | TPH Requester Control Register (Offset 08h) | 1434 |
| 7.9.13.4 | TPH ST Table (Starting from Offset 0Ch) | 1435 |
| 7.9.14 | DPC Extended Capability | 1436 |
| 7.9.14.1 | DPC Extended Capability Header (Offset 00h) | 1439 |
| 7.9.14.2 | DPC Capability Register (Offset 04h) | 1439 |
| 7.9.14.3 | DPC Control Register (Offset 06h) | 1441 |
| 7.9.14.4 | DPC Status Register (Offset 08h) | 1443 |
| 7.9.14.5 | DPC Error Source ID Register (Offset 0Ah) | 1444 |
| 7.9.14.6 | RP PIO Status Register (Offset 0Ch) | 1445 |
| 7.9.14.7 | RP PIO Mask Register (Offset 10h) | 1445 |
| 7.9.14.8 | RP PIO Severity Register (Offset 14h) | 1446 |
| 7.9.14.9 | RP PIO SysError Register (Offset 18h) | 1447 |
| 7.9.14.10 | RP PIO Exception Register (Offset 1Ch) | 1448 |
| 7.9.14.11 | RP PIO Header Log Register (Offset 20h) | 1449 |
| 7.9.14.12 | RP PIO ImpSpec Log Register (Offset 30h) | 1450 |
| 7.9.14.13 | RP PIO TLP Prefix Log Register (Offset 34h) | 1450 |
| 7.9.15 | Precision Time Measurement Extended Capability (PTM Extended Capability) | 1451 |
| 7.9.15.1 | PTM Extended Capability Header (Offset 00h) | 1452 |

| | | |
|---------------|--|-------------|
| 7.9.15.2 | PTM Capability Register (Offset 04h) | 1452 |
| 7.9.15.3 | PTM Control Register (Offset 08h) | 1454 |
| 7.9.16 | Readiness Time Reporting Extended Capability..... | 1455 |
| 7.9.16.1 | Readiness Time Reporting Extended Capability Header (Offset 00h) | 1456 |
| 7.9.16.2 | Readiness Time Reporting 1 Register (Offset 04h)..... | 1457 |
| 7.9.16.3 | Readiness Time Reporting 2 Register (Offset 08h)..... | 1458 |
| 7.9.17 | Hierarchy ID Extended Capability..... | 1459 |
| 7.9.17.1 | Hierarchy ID Extended Capability Header (Offset 00h)..... | 1460 |
| 7.9.17.2 | Hierarchy ID Status Register (Offset 04h) | 1461 |
| 7.9.17.3 | Hierarchy ID Data Register (Offset 08h) | 1462 |
| 7.9.17.4 | Hierarchy ID GUID 1 Register (Offset 0Ch) | 1463 |
| 7.9.17.5 | Hierarchy ID GUID 2 Register (Offset 10h) | 1463 |
| 7.9.17.6 | Hierarchy ID GUID 3 Register (Offset 14h) | 1464 |
| 7.9.17.7 | Hierarchy ID GUID 4 Register (Offset 18h) | 1464 |
| 7.9.17.8 | Hierarchy ID GUID 5 Register (Offset 1Ch) | 1465 |
| 7.9.18 | Vital Product Data Capability (VPD Capability) | 1465 |
| 7.9.18.1 | VPD Address Register | 1466 |
| 7.9.18.2 | VPD Data Register | 1467 |
| 7.9.19 | Native PCIe Enclosure Management Extended Capability (NPEM Extended Capability) | 1467 |
| 7.9.19.1 | NPEM Extended Capability Header (Offset 00h) | 1468 |
| 7.9.19.2 | NPEM Capability Register (Offset 04h) | 1468 |
| 7.9.19.3 | NPEM Control Register (Offset 08h)..... | 1470 |
| 7.9.19.4 | NPEM Status Register (Offset 0Ch) | 1472 |
| 7.9.20 | Alternate Protocol Extended Capability | 1473 |
| 7.9.20.1 | Alternate Protocol Extended Capability Header (Offset 00h) | 1473 |
| 7.9.20.2 | Alternate Protocol Capabilities Register (Offset 04h) | 1474 |
| 7.9.20.3 | Alternate Protocol Control Register (Offset 08h) | 1474 |
| 7.9.20.4 | Alternate Protocol Data 1 Register (Offset 0Ch) | 1475 |
| 7.9.20.5 | Alternate Protocol Data 2 Register (Offset 10h) | 1476 |
| 7.9.20.6 | Alternate Protocol Selective Enable Mask Register (Offset 14h) | 1476 |
| 7.9.21 | Conventional PCI Advanced Features Capability (AF) | 1477 |
| 7.9.21.1 | Advanced Features Capability Header (Offset 00h) | 1477 |
| 7.9.21.2 | AF Capabilities Register (Offset 03h) | 1478 |
| 7.9.21.3 | Conventional PCI Advanced Features Control Register (Offset 04h) | 1478 |
| 7.9.21.4 | AF Status Register (Offset 05h)..... | 1479 |
| 7.9.22 | SFI Extended Capability | 1479 |
| 7.9.22.1 | SFI Extended Capability Header (Offset 00h) | 1480 |
| 7.9.22.2 | SFI Capability Register (Offset 04h) | 1481 |
| 7.9.22.3 | SFI Control Register (Offset 06h) | 1482 |
| 7.9.22.4 | SFI Status Register (Offset 08h)..... | 1485 |
| 7.9.22.5 | SFI CAM Address Register (Offset 0Ch) | 1486 |
| 7.9.22.6 | SFI CAM Data Register (Offset 10h) | 1486 |
| 7.9.23 | Subsystem ID and Subsystem Vendor ID Capability..... | 1487 |
| 7.9.23.1 | Subsystem ID and Subsystem Vendor ID Capability Header (Offset 00h) | 1488 |
| 7.9.23.2 | Subsystem ID and Subsystem Vendor ID Capability Data (Offset 04h) | 1488 |
| 7.9.24 | Data Object Exchange Extended Capability..... | 1488 |
| 7.9.24.1 | DOE Extended Capability Header (Offset 00h) | 1489 |
| 7.9.24.2 | DOE Capabilities Register (Offset 04h) | 1490 |
| 7.9.24.3 | DOE Control Register (Offset 08h) | 1491 |
| 7.9.24.4 | DOE Status Register (Offset 0Ch) | 1492 |
| 7.9.24.5 | DOE Write Data Mailbox Register (Offset 10h)..... | 1493 |

Base 6.4 vs Base 6.3

| | | |
|-------------------|---|-------------|
| 7.9.24.6 | DOE Read Data Mailbox Register (Offset 14h) | 1493 |
| 7.9.25 | Shadow Functions Extended Capability | 1494 |
| 7.9.25.1 | Shadow Functions Extended Capability Header (Offset 00h) | 1496 |
| 7.9.25.2 | Shadow Functions Capability Register (Offset 04h) | 1497 |
| 7.9.25.3 | Shadow Functions Control Register (Offset 08h)..... | 1497 |
| 7.9.25.4 | Shadow Functions Instance Register Entry..... | 1498 |
| 7.9.26 | IDE Extended Capability | 1498 |
| 7.9.26.1 | IDE Extended Capability Header (Offset 00h) | 1499 |
| 7.9.26.2 | IDE Capability Register (Offset 04h)..... | 1500 |
| 7.9.26.3 | IDE Control Register (Offset 08h) | 1502 |
| 7.9.26.4 | Link IDE Register Block | 1502 |
| 7.9.26.4.1 | Link IDE Stream Control Register | 1502 |
| 7.9.26.4.2 | Link IDE Stream Status Register | 1505 |
| 7.9.26.5 | Selective IDE Stream Register Block..... | 1505 |
| 7.9.26.5.1 | Selective IDE Stream Capability Register | 1505 |
| 7.9.26.5.2 | Selective IDE Stream Control Register | 1506 |
| 7.9.26.5.3 | Selective IDE Stream Status Register | 1509 |
| 7.9.26.5.4 | Selective IDE RID Association Register Block | 1509 |
| 7.9.26.5.4.1 | IDE RID Association Register 1..... | 1510 |
| 7.9.26.5.4.2 | IDE RID Association Register 2 | 1510 |
| 7.9.26.5.5 | Selective IDE Address Association Register Block | 1511 |
| 7.9.26.5.5.1 | IDE Address Association Register 1 | 1511 |
| 7.9.26.5.5.2 | IDE Address Association Register 2 | 1511 |
| 7.9.26.5.5.3 | IDE Address Association Register 3 | 1512 |
| 7.9.27 | Null Capability | 1512 |
| 7.9.28 | Null Extended Capability | 1513 |
| 7.9.29 | Streamlined Virtual Channel Extended Capability (SVC) | 1513 |
| 7.9.29.1 | Streamlined Virtual Channel Extended Capability Header (Offset 00h) | 1514 |
| 7.9.29.2 | SVC Port Capability Register 1 (Offset 04h) | 1515 |
| 7.9.29.3 | SVC Port Capability Register 2 (Offset 08h) | 1515 |
| 7.9.29.4 | SVC Port Control Register (Offset 0Ch)..... | 1515 |
| 7.9.29.5 | SVC Port Status Register (Offset 10h) | 1516 |
| 7.9.29.6 | SVC Resource Capability Register..... | 1516 |
| 7.9.29.7 | SVC Resource Control Register | 1517 |
| 7.9.29.8 | SVC Resource Status Register | 1519 |
| 7.9.30 | MMIO Register Block Locator Extended Capability (MRBL) | 1520 |
| 7.9.30.1 | MRBL Extended Capability Header (Offset 00h)..... | 1520 |
| 7.9.30.2 | MRBL Capabilities Register (Offset 04h)..... | 1521 |
| 7.9.30.3 | MRBL Locator Register (Offset Varies) | 1522 |
| 8. | Electrical Sub-Block | 1523 |
| 8.1 | Electrical Specification Introduction..... | 1523 |
| 8.2 | Interoperability Criteria | 1523 |
| 8.2.1 | Data Rates | 1523 |
| 8.2.2 | Refclk Architectures | 1523 |
| 8.3 | Transmitter Specification..... | 1524 |
| 8.3.1 | Measurement Setup for Characterizing Transmitters | 1524 |
| 8.3.1.1 | Breakout and Replica Channels | 1525 |
| 8.3.2 | Voltage Level Definitions | 1525 |
| 8.3.3 | Tx Voltage Parameters | 1526 |
| 8.3.3.1 | 2.5 and 5.0 GT/s Transmitter Equalization | 1526 |

| | | |
|------------|--|------|
| 8.3.3.2 | 8.0, 16.0, 32.0, and 64.0 GT/s Transmitter Equalization | 1526 |
| 8.3.3.3 | Tx Equalization Presets for 8.0, 16.0, 32.0, and 64.0 GT/s | 1528 |
| 8.3.3.4 | Measuring Tx Equalization for 2.5 GT/s and 5.0 GT/s | 1530 |
| 8.3.3.5 | Measuring Presets at 8.0, 16.0, 32.0, and 64.0 GT/s | 1530 |
| 8.3.3.6 | Method for Measuring $V_{TX-DIFF-PP}$ at 2.5 GT/s and 5.0 GT/s..... | 1531 |
| 8.3.3.7 | Method for Measuring $V_{TX-DIFF-PP}$ at 8.0, 16.0, 32.0, and 64.0 GT/s | 1531 |
| 8.3.3.8 | Coefficient Range and Tolerance for 8.0, 16.0, 32.0, and 64.0 GT/s..... | 1532 |
| 8.3.3.9 | EIEOS and $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ Limits | 1535 |
| 8.3.3.10 | Reduced Swing Signaling | 1536 |
| 8.3.3.11 | Effective Tx Package Loss at 8.0, 16.0, 32.0, and 64.0 GT/s | 1536 |
| 8.3.3.12 | Linear Fit Pulse Response | 1538 |
| 8.3.3.13 | Transmitter Signal-to Noise and Distortion Ratio (SNDR _{TX}) for 64.0 GT/s..... | 1540 |
| 8.3.3.14 | Transmitter Ratio of Level Mismatch (R_{LM-TX}) for 64.0 GT/s..... | 1542 |
| 8.3.3.14.1 | Multi Pulse Response Fit (MPRF) – PAM4 Voltage Variant | 1543 |
| 8.3.4 | Transmitter Margining..... | 1547 |
| 8.3.5 | Tx Jitter Parameters | 1548 |
| 8.3.5.1 | Post Processing Steps to Extract Jitter | 1548 |
| 8.3.5.2 | Applying CTLE or De-embedding | 1548 |
| 8.3.5.3 | Independent Refclk Measurement and Post Processing..... | 1549 |
| 8.3.5.4 | Embedded and Non-Embedded Refclk Measurement and Post Processing | 1549 |
| 8.3.5.5 | Behavioral CDR Characteristics | 1550 |
| 8.3.5.6 | Data Dependent and Uncorrelated Jitter..... | 1555 |
| 8.3.5.7 | Data Dependent Jitter..... | 1555 |
| 8.3.5.8 | Uncorrelated Total Jitter and Deterministic Jitter (Dual Dirac Model) (T_{TX-UTJ} and $T_{TX-UDJDD}$) .. | 1556 |
| 8.3.5.9 | Random Jitter (T_{TX-RJ}) (informative) | 1556 |
| 8.3.5.10 | Uncorrelated Total and Deterministic PWJ ($T_{TX-UPW-TJ}$ and $T_{TX-UPW-DJDD}$) .. | 1556 |
| 8.3.6 | Data Rate Dependent Parameters | 1558 |
| 8.3.7 | Tx and Rx Return Loss for 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s | 1562 |
| 8.3.8 | Tx and Rx Return Loss for 64.0 GT/s..... | 1563 |
| 8.3.9 | Transmitter PLL Bandwidth and Peaking | 1565 |
| 8.3.9.1 | 2.5 GT/s and 5.0 GT/s Tx PLL Bandwidth and Peaking | 1565 |
| 8.3.9.2 | 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s Tx PLL Bandwidth and Peaking | 1565 |
| 8.3.9.3 | Series Capacitors..... | 1566 |
| 8.3.10 | Data Rate Independent Tx Parameters..... | 1566 |
| 8.4 | Receiver Specifications | 1567 |
| 8.4.1 | Receiver Stressed Eye Specification | 1567 |
| 8.4.1.1 | Breakout and Replica Channels | 1568 |
| 8.4.1.2 | Calibration Channel Insertion Loss Characteristics..... | 1568 |
| 8.4.1.3 | Post Processing Procedures | 1577 |
| 8.4.1.4 | Behavioral Rx Package Models | 1578 |
| 8.4.1.5 | Behavioral CDR Model..... | 1578 |
| 8.4.1.6 | No Behavioral Rx Equalization for 2.5 and 5.0 GT/s | 1578 |
| 8.4.1.7 | Behavioral Rx Equalization for 8.0, 16.0, 32.0, and 64.0 GT/s | 1578 |
| 8.4.1.8 | Behavioral CTLE (8.0 and 16.0 GT/s) | 1579 |
| 8.4.1.9 | Behavioral CTLE (32.0 and 64.0 GT/s) | 1581 |
| 8.4.1.10 | Behavioral DFE (8.0, 16.0, 32.0, and 64.0 GT/s Only)..... | 1584 |
| 8.4.2 | Stressed Eye Test | 1586 |
| 8.4.2.1 | Procedure for Calibrating a Stressed EH/EW Eye..... | 1586 |
| 8.4.2.1.1 | Post Processing Tool Requirements | 1591 |
| 8.4.2.2 | Procedure for Testing Rx DUT | 1592 |

| | | |
|-----------|---|------|
| 8.4.2.2.1 | Sj Mask..... | 1592 |
| 8.4.2.3 | Receiver Refclk Modes | 1600 |
| 8.4.2.3.1 | Common Refclk Mode | 1600 |
| 8.4.2.3.2 | Independent Refclk Mode | 1601 |
| 8.4.3 | Common Receiver Parameters | 1602 |
| 8.4.3.1 | 5.0 GT/s Exit From Idle Detect (EFI)..... | 1604 |
| 8.4.3.2 | Receiver Return Loss..... | 1604 |
| 8.4.4 | Lane Margining at the Receiver - Electrical Requirements..... | 1604 |
| 8.4.5 | Low Frequency and Miscellaneous Signaling Requirements..... | 1608 |
| 8.4.5.1 | ESD Standards..... | 1608 |
| 8.4.5.2 | Channel AC Coupling Capacitors..... | 1608 |
| 8.4.5.3 | Short Circuit Requirements | 1608 |
| 8.4.5.4 | Transmitter and Receiver Termination..... | 1609 |
| 8.4.5.5 | Electrical Idle | 1609 |
| 8.4.5.6 | DC Common Mode Voltage | 1609 |
| 8.4.5.7 | Receiver Detection | 1610 |
| 8.5 | Channel Tolerancing | 1610 |
| 8.5.1 | Channel Compliance Testing | 1610 |
| 8.5.1.1 | Behavioral Transmitter and Receiver Package Models..... | 1612 |
| 8.5.1.2 | Measuring Package Performance (16.0 GT/s only)..... | 1622 |
| 8.5.1.3 | Simulation Tool Requirements..... | 1622 |
| 8.5.1.3.1 | Simulation Tool Chain Inputs | 1623 |
| 8.5.1.3.2 | Processing Steps | 1623 |
| 8.5.1.3.3 | Simulation Tool Outputs..... | 1623 |
| 8.5.1.3.4 | Open Source Simulation Tool..... | 1624 |
| 8.5.1.4 | Behavioral Transmitter Parameters | 1624 |
| 8.5.1.4.1 | Deriving Voltage and Jitter Parameters | 1624 |
| 8.5.1.4.2 | Optimizing Tx/Rx Equalization (8.0, 16.0, 32.0, and 64.0 GT/s only) | 1626 |
| 8.5.1.4.3 | Pass/Fail Eye Characteristics | 1626 |
| 8.5.1.4.4 | Characterizing Channel Common Mode Noise..... | 1629 |
| 8.5.1.4.5 | Verifying VCH-IDLE-DET-DIFF-pp | 1630 |
| 8.6 | Refclk Specifications | 1630 |
| 8.6.1 | Refclk Test Setup | 1630 |
| 8.6.2 | REFCLK AC Specifications | 1631 |
| 8.6.3 | Data Rate Independent Refclk Parameters | 1634 |
| 8.6.3.1 | Low Frequency Refclk Jitter Limits | 1635 |
| 8.6.4 | Refclk Architectures Supported..... | 1636 |
| 8.6.5 | Filtering Functions Applied to Raw Data | 1636 |
| 8.6.5.1 | PLL Filter Transfer Function Example..... | 1637 |
| 8.6.5.2 | CDR Transfer Function Examples..... | 1637 |
| 8.6.6 | Common Refclk Rx Architecture (CC) | 1638 |
| 8.6.6.1 | Determining the Number of PLL BW and peaking Combinations..... | 1639 |
| 8.6.6.2 | CDR and PLL BW and Peaking Limits for Common Refclk..... | 1639 |
| 8.6.7 | Jitter Limits for Refclk Architectures | 1640 |
| 8.6.8 | Form Factor Requirements for RefClock Architectures | 1641 |
| 9. | I/O Virtualization and Sharing | 1643 |
| 9.1 | ↑↑IOV↑ Architectural Overview..... | 1643 |

ECN: Base 6.3
SIOV△↔

9.1.1

Base 6.4 vs Base 6.3

| | | |
|--|------|-------------------------|
| ↑↑RID in Endpoints where a PF contains an SIOV Extended Capability Structure↑ | 1656 | ECN: Base 6.3 SIOV△↔ |
| 9.1.2 ↑↑SDI Reset↑ | 1657 | ECN: Base 6.3 SIOV△↔ |
| 9.2 SR-IOV Initialization and Resource Allocation | 1658 | |
| 9.2.1 SR-IOV Resource Discovery | 1658 | |
| 9.2.1.1 Configuring SR-IOV Capabilities | 1658 | |
| 9.2.1.1.1 Configuring the VF BAR Mechanisms | 1659 | |
| 9.2.1.2 VF Discovery | 1660 | |
| 9.2.1.3 Function Dependency Lists..... | 1663 | |
| 9.2.1.4 Interrupt Resource Allocation..... | 1663 | |
| 9.2.2 SR-IOV Reset Mechanisms..... | 1663 | |
| 9.2.2.1 SR-IOV Conventional Reset..... | 1663 | |
| 9.2.2.2 FLR That Targets a VF | 1663 | |
| 9.2.2.3 FLR That Targets a PF | 1663 | |
| 9.2.3 IOV Re-initialization and Reallocation..... | 1664 | |
| 9.3 ↑↑SIOV Initialization and Resource Allocation↑ | 1664 | ECN: Base 6.3 SIOV△↔ |
| 9.3.1 ↑↑SIOV Resource Discovery↑ | 1664 | |
| 9.3.1.1 ↑↑Configuring SIOV Capabilities↑ | 1664 | |
| 9.3.1.1.1 ↑↑Configuring an SDI↑ | 1664 | |
| 9.3.1.1.1.1 ↑↑Behavior of a VI↑ | 1664 | ECN: Base 6.3 SIOV△↔ |
| 9.3.1.1.1.2 ↑↑Requirements of an SI↑ | 1665 | ECN: Base 6.3 SIOV△↔ |
| 9.3.1.2 ↑↑MSI/MSI-X Interrupt Message Resource Allocation↑ | 1665 | |
| 9.3.1.2 ↑↑SR-IOV↑ Configuration | 1666 | |
| 9.4.1 SR-IOV Configuration Overview | 1666 | |
| 9.4.2 ↑↑SR-IOV↑ Configuration Space | 1666 | |
| 9.4.3 SR-IOV Extended Capability | 1666 | |
| 9.4.3.1 SR-IOV Extended Capability Header (Offset 00h) | 1667 | |
| 9.4.3.2 SR-IOV Capabilities Register (04h) | 1668 | |
| 9.4.3.2.1 VF Migration Capable..... | 1669 | |
| 9.4.3.2.2 ARI Capable Hierarchy Preserved | 1669 | |
| 9.4.3.2.3 VF Larger-Tag Requester Support..... | 1669 | |
| 9.4.3.2.4 VF Migration Interrupt Message Number | 1670 | |
| 9.4.3.3 SR-IOV Control Register (Offset 08h) | 1670 | |
| 9.4.3.3.1 VF Enable..... | 1672 | |
| 9.4.3.3.2 VF Migration Enable..... | 1673 | |
| 9.4.3.3.3 VF Migration Interrupt Enable | 1673 | |
| 9.4.3.3.4 VF MSE (Memory Space Enable) | 1673 | |
| 9.4.3.3.5 ARI Capable Hierarchy | 1674 | |
| 9.4.3.4 SR-IOV Status Register (Offset 0Ah) | 1674 | |
| 9.4.3.4.1 VF Migration Status..... | 1675 | |
| 9.4.3.5 InitialVFs (Offset 0Ch)..... | 1675 | |
| 9.4.3.6 TotalVFs (Offset 0Eh) | 1675 | |
| 9.4.3.7 NumVFs (Offset 10h)..... | 1675 | |
| 9.4.3.8 Function Dependency Link (Offset 12h)..... | 1675 | |
| 9.4.3.9 First VF Offset (Offset 14h) | 1678 | |

Base 6.4 vs Base 6.3

| | | |
|-----------|---|-------------------------|
| 9.4.3.10 | VF Stride (Offset 16h) | 1678 |
| 9.4.3.11 | VF Device ID (Offset 1Ah) | 1678 |
| 9.4.3.12 | Supported Page Sizes (Offset 1Ch) | 1679 |
| 9.4.3.13 | System Page Size (Offset 20h)..... | 1679 |
| 9.4.3.14 | VF BAR0 (Offset 24h), VF BAR1 (Offset 28h), VF BAR2 (Offset 2Ch), VF BAR3 (Offset 30h), VF BAR4 (Offset 34h), VF BAR5 (Offset 38h)..... | 1679 |
| 9.4.3.15 | VF Migration State Array Offset (Deprecated) (Offset 3Ch) | 1680 |
| 9.4.4 | PF/VF Configuration Space Header | 1680 |
| 9.4.5 | PCI Express Capability Changes | 1680 |
| 9.4.6 | PCI Standard Capabilities | 1680 |
| 9.4.7 | PCI Express Extended Capabilities Changes | 1681 |
| 9.5 | ↑↑SIOV Configuration↑↑ | 1684 |
| 9.5.1 | ↑↑SIOV Configuration Overview↑↑ | 1684 |
| | | ECN: Base 6.3 SIOV△↔ |
| 9.5.2 | ↑↑SIOV Configuration Space↑↑ | 1685 |
| 9.5.3 | ↑↑SIOV Extended Capability↑↑ | 1685 |
| 9.5.4 | ↑↑SIOV Extended Capability Header (Offset 00h)↑↑ | 1685 |
| 9.5.5 | ↑↑SIOV Capabilities Register (04h)↑↑ | 1686 |
| 9.5.6 | ↑↑Total SDIs (Offset 08h)↑↑ | 1686 |
| 9.5.7 | ↑↑SIOV Status Register (Offset 0Bh)↑↑ | 1687 |
| 9.5.8 | ↑↑First SDI Offset (Offset 0Ch)↑↑ | 1687 |
| 9.5.9 | ↑↑SDI Stride (Offset 0Eh)↑↑ | 1687 |
| 10. | Address Translation Services (ATS) | 1689 |
| 10.1 | ATS Architectural Overview | 1689 |
| 10.1.1 | Address Translation Services (ATS) Overview | 1690 |
| 10.1.2 | Page Request Interface Extension | 1694 |
| 10.1.3 | Process Address Space ID (PASID) | 1695 |
| 10.1.4 | ATS Memory Attributes | 1696 |
| 10.2 | ATS Translation Services | 1696 |
| 10.2.1 | Memory Requests with Address Type | 1696 |
| 10.2.2 | Translation Requests | 1698 |
| 10.2.2.1 | Attribute Field..... | 1700 |
| 10.2.2.2 | Length Field..... | 1700 |
| 10.2.2.3 | Tag Field..... | 1700 |
| 10.2.2.4 | Untranslated Address Field | 1701 |
| 10.2.2.5 | No Write (NW) Flag..... | 1701 |
| 10.2.2.6 | PASID on Translation Request | 1701 |
| 10.2.2.7 | CXL Src | 1701 |
| 10.2.3 | Translation Completion | 1702 |
| 10.2.3.1 | Translated Address Field..... | 1705 |
| 10.2.3.2 | Translation Range Size (S) Field | 1706 |
| 10.2.3.3 | Non-snooped (N) Field..... | 1706 |
| 10.2.3.4 | Untranslated Access Only (U) Field | 1707 |
| 10.2.3.5 | Read (R) and Write (W) Fields..... | 1707 |
| 10.2.3.6 | Execute Permitted (Exe) | 1708 |
| 10.2.3.7 | Privileged Mode Access (Priv) | 1709 |
| 10.2.3.8 | Global Mapping (Global) | 1709 |
| 10.2.3.9 | ATS Memory Attributes | 1712 |
| 10.2.3.10 | ↑↑TEE Exclusive Memory Attribute (TE)↑↑ | 1715 |
| | | ECN: Base 6.3 XT△↔ |

| | | |
|-------------------|---|-------------|
| 10.2.4 | Completions with Multiple Translations | 1715 |
| 10.3 | ATS Invalidation..... | 1716 |
| 10.3.1 | Invalidate Request..... | 1716 |
| 10.3.2 | Invalidate Completion | 1718 |
| 10.3.3 | Invalidate Completion Semantics | 1720 |
| 10.3.4 | Request Acceptance Rules | 1721 |
| 10.3.5 | Invalidate Flow Control..... | 1722 |
| 10.3.6 | Invalidate Ordering Semantics | 1722 |
| 10.3.7 | Implicit Invalidation Events | 1723 |
| 10.3.8 | PASID and Global Invalidate | 1724 |
| 10.4 | Page Request Services | 1724 |
| 10.4.1 | Page Request Message | 1725 |
| 10.4.1.1 | PASID Usage..... | 1727 |
| 10.4.1.2 | Managing PASID Usage on PRG Requests | 1727 |
| 10.4.1.2.1 | Stop Marker Messages..... | 1728 |
| 10.4.2 | Page Request Group Response Message..... | 1730 |
| 10.4.2.1 | Response Code Field..... | 1731 |
| 10.4.2.2 | PASID Usage on PRG Responses | 1732 |
| 10.5 | ATS Configuration..... | 1732 |
| 10.5.1 | ATS Extended Capability | 1732 |
| 10.5.1.1 | ATS Extended Capability Header (Offset 00h) | 1733 |
| 10.5.1.2 | ATS Capability Register (Offset 04h) | 1733 |
| 10.5.1.3 | ATS Control Register (Offset 06h)..... | 1735 |
| 10.5.2 | Page Request Extended Capability Structure | 1736 |
| 10.5.2.1 | Page Request Extended Capability Header (Offset 00h)..... | 1736 |
| 10.5.2.2 | Page Request Control Register (Offset 04h) | 1737 |
| 10.5.2.3 | Page Request Status Register (Offset 06h) | 1738 |
| 10.5.2.4 | Outstanding Page Request Capacity (Offset 08h) | 1739 |
| 10.5.2.5 | Outstanding Page Request Allocation (Offset 0Ch) | 1739 |
| 11. | TEE Device Interface Security Protocol (TDISP) | 1741 |
| 11.1 | Overview of the TEE-I/O Security Model as it Relates to Devices..... | 1743 |
| 11.2 | TDISP Rules..... | 1748 |
| 11.2.1 | TDISP TLP Rules | 1752 |
| 11.2.2 | TDISP Message Transport | 1755 |
| 11.2.3 | Requirements for Requesters (TSM) | 1756 |
| 11.2.4 | Requirements for Responders (DSM) | 1756 |
| 11.2.5 | TDISP Timing Requirements..... | 1757 |
| 11.2.6 | DSM Tracking and Handling of Locked TDI Configurations (Informative) | 1757 |
| 11.2.7 | TVM Acceptance of a TDI | 1760 |
| 11.3 | TDISP Message Formats and processing..... | 1760 |
| 11.3.1 | TDISP Request Codes | 1760 |
| 11.3.2 | TDISP Response Codes | 1761 |
| 11.3.3 | TDISP Message Format and Protocol Versioning..... | 1762 |
| 11.3.4 | GET_TDISP_VERSION..... | 1763 |
| 11.3.5 | TDISP_VERSION..... | 1763 |
| 11.3.6 | GET_TDISP_CAPABILITIES | 1764 |
| 11.3.7 | TDISP_CAPABILITIES..... | 1764 |
| 11.3.8 | LOCK_INTERFACE_REQUEST..... | 1765 |
| 11.3.9 | LOCK_INTERFACE_RESPONSE | 1767 |
| 11.3.10 | GET_DEVICE_INTERFACE_REPORT | 1768 |

Base 6.4 vs Base 6.3

| | | |
|----------|--|--------------------|
| 11.3.11 | DEVICE_INTERFACE_REPORT | 1769 |
| 11.3.12 | GET_DEVICE_INTERFACE_STATE..... | 1771 |
| 11.3.13 | DEVICE_INTERFACE_STATE..... | 1772 |
| 11.3.14 | START_INTERFACE_REQUEST | 1772 |
| 11.3.15 | START_INTERFACE_RESPONSE | 1772 |
| 11.3.16 | STOP_INTERFACE_REQUEST | 1773 |
| 11.3.17 | STOP_INTERFACE_RESPONSE..... | 1773 |
| 11.3.18 | BIND_P2P_STREAM_REQUEST | 1773 |
| 11.3.19 | BIND_P2P_STREAM_RESPONSE | 1774 |
| 11.3.20 | UNBIND_P2P_STREAM_REQUEST..... | 1775 |
| 11.3.21 | UNBIND_P2P_STREAM_RESPONSE | 1776 |
| 11.3.22 | SET_MMIO_ATTRIBUTE_REQUEST | 1776 |
| 11.3.23 | SET_MMIO_ATTRIBUTE_RESPONSE | 1777 |
| 11.3.24 | TDISP_ERROR..... | 1778 |
| 11.3.25 | VDM_REQUEST | 1779 |
| 11.3.26 | VDM_RESPONSE..... | 1779 |
| 11.3.27 | ↑↑SET_TDISP_CONFIG_REQUEST↑↑..... | 1780 |
| | | ECN: Base 6.3 XT△↔ |
| 11.3.28 | ↑↑SET_TDISP_CONFIG_RESPONSE↑↑..... | 1780 |
| | | ECN: Base 6.3 XT△↔ |
| 11.4 | Device Security Requirements..... | 1780 |
| 11.4.1 | Device Identity and Authentication..... | 1780 |
| 11.4.2 | Firmware and Configuration Measurements | 1781 |
| 11.4.3 | Securing Interconnects | 1781 |
| 11.4.4 | Device Attached Memory | 1782 |
| 11.4.5 | TDI Security | 1782 |
| 11.4.6 | Data Integrity Errors | 1783 |
| 11.4.7 | Debug Modes..... | 1783 |
| 11.4.8 | Conventional Reset | 1784 |
| 11.4.9 | Function Level Reset | 1784 |
| 11.4.10 | Address Translation Services (ATS) and Access Control Services (ACS)..... | 1784 |
| 11.5 | Requirements Placed on Host Security due to TDI Requirements | 1788 |
| 11.5.1 | Address Translation..... | 1788 |
| 11.5.2 | MMIO Access Control..... | 1788 |
| 11.5.3 | DMA Access Control..... | 1789 |
| 11.5.4 | Device Binding..... | 1789 |
| 11.5.5 | Securing Interconnects | 1790 |
| 11.5.6 | Data Integrity Errors | 1790 |
| 11.5.7 | TSM Tracking and Handling of Locked Root Port Configurations (Informative) | 1790 |
| 11.5.8 | IDE Extended Capability registers..... | 1793 |
| 11.6 | Overview of Threat Model and Mitigations | 1794 |
| 11.6.1 | Interconnect Security..... | 1794 |
| 11.6.2 | Identity and Measurement Reporting | 1795 |
| 11.6.3 | TDI Assignment and Detach..... | 1795 |
| 12. | Architectural Out-of-Band Management..... | 1797 |
| 12.1 | Introduction..... | 1797 |
| 12.2 | Framework for Sidebands..... | 1797 |
| 12.3 | Sideband Signaling Mechanisms..... | 1798 |
| 12.3.1 | Discrete Sidebands..... | 1798 |
| 12.3.2 | Flex I/O Sidebands | 1799 |
| 12.3.2.1 | Flex I/O Default State Guidelines | 1799 |

Base 6.4 vs Base 6.3

| | | |
|---------------|---|-------------|
| 12.3.2.2 | Flex I/O Discovery Phase Guidelines | 1800 |
| 12.3.2.3 | Flex I/O Compatibility Check Guidelines..... | 1800 |
| 12.3.2.4 | Flex I/O Control Negotiation Guidelines | 1801 |
| 12.3.2.5 | General Flex I/O Control Guidelines | 1801 |
| 12.3.3 | Peripheral Sideband Tunnelling Interface (PESTI) Sidebands | 1802 |
| 12.3.3.1 | PESTI Introduction | 1802 |
| 12.3.3.2 | PESTI Physical Interface..... | 1803 |
| 12.3.3.3 | PESTI Electrical Circuit..... | 1803 |
| 12.3.3.4 | PESTI DC Specifications | 1805 |
| 12.3.3.5 | PESTI Target Detection | 1805 |
| 12.3.3.6 | PESTI Protocol Commands..... | 1806 |
| 12.3.3.6.1 | Discovery Payload Request (DPR) | 1806 |
| 12.3.3.6.2 | PESTI Virtual Wire Exchange (VWE) | 1806 |
| 12.3.3.6.3 | PESTI Fanout MUX Control | 1806 |
| 12.3.3.7 | PESTI Initiator Abort | 1807 |
| 12.3.3.8 | PESTI Broadcast | 1807 |
| 12.3.3.9 | PESTI Initiator Control and Status Registers..... | 1808 |
| 12.3.3.10 | PESTI AC Specifications | 1810 |
| 12.3.3.11 | PESTI Discovery Phase..... | 1811 |
| 12.3.3.12 | PESTI Active Phase | 1814 |
| 12.3.3.13 | PESTI Target Reset and Fault Handling..... | 1816 |
| 12.3.3.14 | PESTI Fan-Out..... | 1816 |
| 12.3.3.15 | PESTI Security Considerations | 1820 |
| 12.4 | Managed USB 2.0..... | 1820 |
| 12.5 | 2-Wire Interface | 1821 |
| 12.5.1 | 2-Wire Interface Use Cases..... | 1822 |
| 12.5.2 | 2-Wire Addressing | 1822 |
| 12.5.3 | ↑↓2-wire↑ ↑↓2-Wire↑ Bus Sharing | 1824 |
| 12.5.3.1 | ↑↓2-wire↑ ↑↓2-Wire↑ Multi-Drop Topology | 1824 |
| 12.5.3.2 | SMBus MUX Use..... | 1824 |
| 12.5.3.3 | ↑↓2-wire↑ ↑↓2-Wire↑ Hub Use | 1825 |
| 12.5.4 | [I3C-Basic] Support on Existing SMBus Signals | 1826 |
| 12.5.4.1 | I3C Basic Overview | 1826 |
| 12.5.4.2 | I3C Basic Discovery and Mode Changing | 1827 |
| 12.5.4.3 | I3C Basic DC and AC Signal Requirements | 1830 |
| 12.6 | Field Replacement Unit (FRU) Information..... | 1832 |
| 12.6.1 | FRU Information Device Requirements | 1832 |
| 12.6.1.1 | FRU Information Device Requirements Specific to SMBus/I2C Mode | 1833 |
| 12.6.1.2 | [SMBus]/[I2C] Access Protocol | 1834 |
| 12.6.2 | FRU Information Format | 1835 |
| 12.6.3 | Common PCI-SIG MultiRecord Descriptors | 1837 |
| 12.6.3.1 | Connector Subdivision (Group ID 0h, Sub-Type 0h) | 1837 |
| 12.7 | Out-of-Band ↑↓Management↑ Control Mechanism | 1840 |
| 12.7.1 | ↑↓PCIe-MI Architectural Model | 1841 |
| 12.7.1.1 | ↑↓Operational Times↑ | 1843 |
| 12.7.1.2 | | |

 ECN: Base 6.3
 PCIe-MI△↔

 ECN: Base 6.3
 PCIe-MI△↔

 ECN: Base 6.3
 PCIe-MI△↔

Base 6.4 vs Base 6.3

| | | |
|---|------|-----------------------------|
| 12.7.1.3 ↑↓PCIe-MI Completer Reset↑ | 1844 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.1.4 ↑↓Example PCIe-MI Component↑ | 1846 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.1.5 ↑↓Example PCIe-MI Management Traffic Flows↑ | 1849 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.2 ↑↓PCIe-MI Message Format↑ | 1851 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.2.1 ↑↓PCIe-MI Opcodes↑ | 1854 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.2.2 ↑↓PCIe-MI Status↑ | 1855 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.2.2.1 ↑↓Invalid Input Error Completion↑ | 1856 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3 ↑↓PCIe-MI Commands↑ | 1857 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1 ↑↓Get Information (Opcode 00h)↑ | 1859 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.1 ↑↓Supported Information Identifiers (Information Identifier 00h)↑ | 1860 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.2 ↑↓Supported Opcodes (Information Identifier 01h)↑ | 1861 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.3 ↑↓Supported Parameters (Information Identifier 02h)↑ | 1861 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.4 ↑↓Component Information (Information Identifier 03h)↑ | 1862 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.5 ↑↓Port Information (Information Identifier 04h)↑ | 1863 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.6 ↑↓Function List (Information Identifier 05h)↑ | 1866 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.1.7 ↑↓Function Information (Information Identifier 06h)↑ | 1868 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.2 ↑↓Get Parameters (Opcode 01h) and Set Parameters (Opcode 02h)↑ | 1870 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.2.1 ↑↓Configuration Space Writability (Parameter Identifier 00h)↑ | 1872 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.2.1.1 ↑↓Get Parameters – Configuration Space Writability↑ | 1873 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.2.1.2 ↑↓Set Parameters – Configuration Space Writability↑ | 1876 | ECN: Base 6.3 PCIe-MI△◀▶ |
| 12.7.3.2.2 ↑↓Routing Information (Parameter Identifier 01h)↑ | 1878 | ECN: Base 6.3 PCIe-MI△◀▶ |

Base 6.4 vs Base 6.3

| | | | |
|--------------|---|-------------|-----------------------------|
| 12.7.3.2.2.1 | ↑↓Get Parameters – Routing Information↑ | 1879 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.2.2 | ↑↓Set Parameters – Routing Information↑ | 1879 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.3 | ↑↓Configuration Space (Parameter Identifier 02h)↑ | 1880 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.3.1 | ↑↓Get Parameters – Configuration Space↑ | 1881 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.3.2 | ↑↓Set Parameters – Configuration Space↑ | 1881 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.4 | ↑↓Connector Subdivision (Parameter Identifier 03h)↑ | 1882 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.4.1 | ↑↓Get Parameters – Connector Subdivision↑ | 1882 | ECN: Base 6.3 PCIe-MI△◀▷ |
| 12.7.3.2.4.2 | ↑↓Set Parameters – Connector Subdivision↑ | 1883 | ECN: Base 6.3 PCIe-MI△◀ |
| 12.8 | Retimer Management..... | 1884 | |
| 12.9 | Internal Cable Management | 1884 | |
| A. | Isochronous Applications | 1887 | |
| A.1 | Introduction..... | 1887 | |
| A.2 | Isochronous Contract and Contract Parameters | 1888 | |
| A.2.1 | Isochronous Time Period and Isochronous Virtual Timeslot..... | 1889 | |
| A.2.2 | Isochronous Payload Size | 1889 | |
| A.2.3 | Isochronous Bandwidth Allocation..... | 1889 | |
| A.2.4 | Isochronous Transaction Latency | 1891 | |
| A.2.5 | An Example Illustrating Isochronous Parameters..... | 1892 | |
| A.3 | Isochronous Transaction Rules | 1892 | |
| A.4 | Transaction Ordering | 1892 | |
| A.5 | Isochronous Data Coherency..... | 1893 | |
| A.6 | Flow Control | 1893 | |
| A.7 | Considerations for Bandwidth Allocation | 1893 | |
| A.7.1 | Isochronous Bandwidth of PCI Express Links..... | 1893 | |
| A.7.2 | Isochronous Bandwidth of Endpoints | 1894 | |
| A.7.3 | Isochronous Bandwidth of Switches..... | 1894 | |
| A.7.4 | Isochronous Bandwidth of Root Complex | 1894 | |
| A.8 | Considerations for PCI Express Components..... | 1894 | |
| A.8.1 | An Endpoint as a Requester..... | 1894 | |
| A.8.2 | An Endpoint as a Completer | 1894 | |
| A.8.3 | Switches..... | 1895 | |
| A.8.4 | Root Complex | 1895 | |
| B. | Symbol Encoding | 1897 | |
| C. | Physical Layer Appendix | 1907 | |
| C.1 | 8b/10b Data Scrambling Example | 1907 | |
| C.2 | 128b/130b Data Scrambling Example | 1912 | |

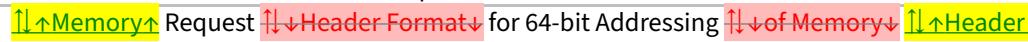
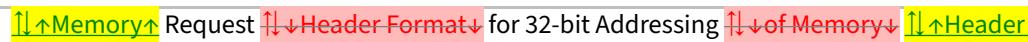
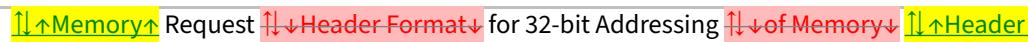
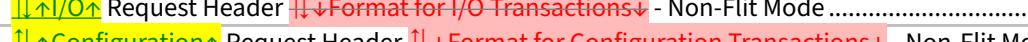
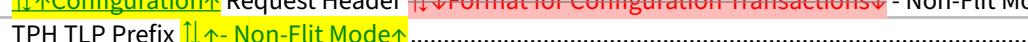
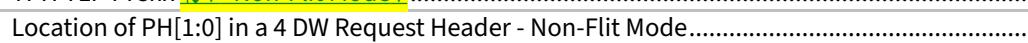
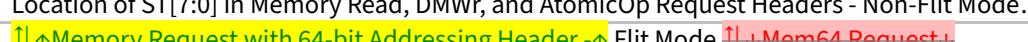
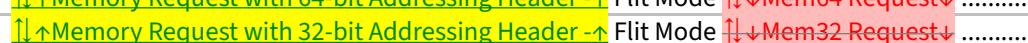
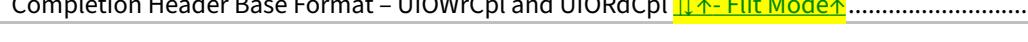
| | | |
|-------|--|------|
| D. | Request Dependencies | 1915 |
| E. | ID-Based Ordering Usage | 1919 |
| E.1 | Introduction..... | 1919 |
| E.2 | Potential Benefits with IDO Use..... | 1920 |
| E.2.1 | Benefits for MFD/RP Direct Connect..... | 1920 |
| E.2.2 | Benefits for Switched Environments..... | 1920 |
| E.2.3 | Benefits for Integrated Endpoints | 1920 |
| E.2.4 | IDO Use in Conjunction with RO | 1921 |
| E.3 | When to Use IDO..... | 1921 |
| E.4 | When Not to Use IDO | 1921 |
| E.4.1 | When Not to Use IDO with Endpoints..... | 1921 |
| E.4.2 | When Not to Use IDO with Root Ports | 1922 |
| E.5 | Software Control of IDO Use | 1922 |
| E.5.1 | Software Control of Endpoint IDO Use | 1922 |
| E.5.2 | Software Control of Root Port IDO Use | 1923 |
| F. | Message Code Usage | 1925 |
| G. | Protocol Multiplexing | 1927 |
| G.1 | Protocol Multiplexing Interactions with PCI Express | 1928 |
| G.2 | PMUX Packets | 1932 |
| G.3 | PMUX Packet Layout | 1933 |
| G.3.1 | PMUX Packet Layout for 8b/10b Encoding..... | 1933 |
| G.3.2 | PMUX Packet Layout at 128b/130b Encoding | 1934 |
| G.4 | PMUX Control | 1937 |
| G.5 | PMUX Extended Capability..... | 1937 |
| G.5.1 | PMUX Extended Capability Header (Offset 00h) | 1938 |
| G.5.2 | PMUX Capability Register (Offset 04h) | 1938 |
| G.5.3 | PMUX Control Register (Offset 08h)..... | 1939 |
| G.5.4 | PMUX Status Register (Offset 0Ch)..... | 1941 |
| G.5.5 | PMUX Protocol Array (Offsets 10h through 48h) | 1942 |
| H. | Flow Control Update Latency and ACK Update Latency Calculations..... | 1945 |
| H.1 | Flow Control Update Latency | 1945 |
| H.2 | Ack Latency..... | 1947 |
| I. | Async Hot-Plug Reference Model | 1951 |
| I.1 | Async Hot-Plug Initial Configuration | 1953 |
| I.2 | Async Removal Configuration and Interrupt Handling | 1954 |
| I.3 | Async Hot-Add Configuration and Interrupt Handling | 1956 |
| J. | Alpha Power and Reverse lookup assignment | 1959 |
| J.1 | Alpha Powers..... | 1960 |
| J.2 | 84 Byte to 86 Byte Encoder | 1970 |
| J.3 | 250 Byte to 256 Byte Encoder example | 1971 |
| J.4 | 86 Byte to 84 Byte Decoder | 1974 |
| J.5 | 256 Byte to 250 Byte decoder | 1978 |
| K. | MATLAB created generator matrix for CRC code | 1983 |
| K.1 | Generator Matrix output | 1984 |

| | | |
|-----|---|------|
| K.2 | Flit 8 byte LCRC..... | 2016 |
| L. | Example IDE TLPs and Test Keys | 2257 |
| L.1 | Example NFM IDE TLP Without Partial Header Encryption | 2257 |
| L.2 | Example FM IDE TLP Without Partial Header Encryption..... | 2259 |
| L.3 | Example NFM IDE TLP With Partial Header Encryption..... | 2263 |
| L.4 | Example FM IDE TLP With Partial Header Encryption..... | 2266 |
| L.5 | IDE Test Keys..... | 2270 |

List of Figures

| | | |
|-------------|--|-----|
| Figure 1 | Old Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Transmit side | 87 |
| Figure 2 | New Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Transmit side | 88 |
| Figure 3 | Old Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Receive side | 88 |
| Figure 4 | New Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Receive side | 89 |
| Figure 5 | Old Figure: Powers of alpha for the check bits for Bytes 0 to 84 | 90 |
| Figure 6 | New Figure: Powers of alpha for the check bits for Bytes 0 to 84 | 90 |
| Figure 1-1 | PCI Express Link | 142 |
| Figure 1-2 | Example PCI Express Topology | 144 |
| Figure 1-3 | Logical Block Diagram of a Switch | 147 |
| Figure 1-4 | High-Level Layering Diagram | 149 |
| Figure 1-5 | Packet Flow Through the Layers | 149 |
| Figure 2-1 | Layering Diagram Highlighting the Transaction Layer | 155 |
| Figure 2-2 | Serial View of a ↑↑Non-Flit Mode↑ TLP | 157 |
| Figure 2-3 | Generic TLP Format - Non-Flit Mode | 158 |
| Figure 2-4 | Fields Present in All ↑↑Non-Flit Mode↑ TLPs | 159 |
| Figure 2-5 | Fields Present in All Non-Flit Mode TLP Headers | 160 |
| Figure 2-6 | First DW of Header Base | 163 |
| Figure 2-7 | OHC-A1 | 175 |
| Figure 2-8 | OHC-A2 | 176 |
| Figure 2-9 | OHC-A3 | 176 |
| Figure 2-10 | OHC-A4 | 177 |
| Figure 2-11 | OHC-A5 | 177 |
| Figure 2-12 | OHC-B | 177 |
| Figure 2-13 | OHC-C | 178 |
| Figure 2-14 | Example Topology Illustrating Multiple Segments and NFM Subtrees | 181 |
| Figure 2-15 | Examples of Completer Target Memory Access for FetchAdd | 188 |
| Figure 2-16 | 32-bit Address Routing - Non-Flit Mode | 191 |
| Figure 2-17 | 64-bit Address Routing - Non-Flit Mode | 192 |
| Figure 2-18 | 32-bit Address Routing - Flit Mode | 193 |
| Figure 2-19 | 64-bit Address Routing - Flit Mode ↑↑- 4 DW↑ | 193 |
| Figure 2-20 | 64-bit Address Routing - Flit Mode - 5 DW | 194 |
| Figure 2-21 | 64-bit Address Routing - Flit Mode - 6 DW | 195 |
| Figure 2-22 | 64-bit Address Routing - Flit Mode - 7 DW | 196 |
| Figure 2-23 | Non-ARI ID Routing with 4 DW Header - Non-Flit Mode | 199 |
| Figure 2-24 | ARI ID Routing with 4 DW Header - Non-Flit Mode | 200 |
| Figure 2-25 | Non-ARI ID Routing with 3 DW Header - Non-Flit Mode | 201 |
| Figure 2-26 | ARI ID Routing with 3 DW Header - Non-Flit Mode | 202 |
| Figure 2-27 | ID Routing with 3 DW Header - Flit Mode | 203 |
| Figure 2-28 | ID Routing with 4 DW Header - Flit Mode | 204 |
| Figure 2-29 | ID Routing with 5 DW Header - Flit Mode | 205 |
| Figure 2-30 | ID Routing with 6 DW Header - Flit Mode | 206 |
| Figure 2-31 | ID Routing with 7 DW Header - Flit Mode | 207 |
| Figure 2-32 | Location of Byte Enables in TLP Header - Non-Flit Mode | 208 |
| Figure 2-33 | Transaction Descriptor | 211 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|-----|
| Figure 2-34 | Transaction ID | 212 |
| Figure 2-35 | Attributes Field of Transaction Descriptor | 220 |
| Figure 2-36 |  for 64-bit Addressing  | 224 |
| Figure 2-37 |  for 32-bit Addressing  | 224 |
| Figure 2-38 |  - Non-Flit Mode | 226 |
| Figure 2-39 |  - Non-Flit Mode ... | 227 |
| Figure 2-40 | TPH TLP Prefix  | 227 |
| Figure 2-41 | Location of PH[1:0] in a 4 DW Request Header - Non-Flit Mode | 229 |
| Figure 2-42 | Location of PH[1:0] in a 3 DW Request Header - Non-Flit Mode | 230 |
| Figure 2-43 | Location of ST[7:0] in the Memory Write Request Header - Non-Flit Mode | 231 |
| Figure 2-44 | Location of ST[7:0] in Memory Read, DMWr, and AtomicOp Request Headers - Non-Flit Mode | 232 |
| Figure 2-45 |  Flit Mode  | 233 |
| Figure 2-46 |  | 233 |
| Figure 2-47 |  | 234 |
| Figure 2-48 |  | 234 |
| Figure 2-49 | Message Request Header - Non-Flit Mode | 236 |
| Figure 2-50 | Message Request Header - Flit Mode..... | 237 |
| Figure 2-51 | ERR_COR Message - Non-Flit Mode | 243 |
| Figure 2-52 | ERR_COR Message - Flit Mode | 244 |
| Figure 2-53 | Header for Vendor-Defined Messages - Non-Flit Mode..... | 247 |
| Figure 2-54 | Header for Vendor-Defined Messages - Flit Mode..... | 247 |
| Figure 2-55 | Header for PCI-SIG-Defined VDMs - Non-Flit Mode..... | 248 |
| Figure 2-56 | Header for PCI-SIG-Defined VDMs - Flit Mode | 249 |
| Figure 2-57 | DRS Message - Non-Flit Mode | 250 |
| Figure 2-58 | DRS Message - Flit Mode | 250 |
| Figure 2-59 | FRS Message - Non-Flit Mode..... | 252 |
| Figure 2-60 | FRS Message - Flit Mode..... | 252 |
| Figure 2-61 | Hierarchy ID Message - Non-Flit Mode | 253 |
| Figure 2-62 | Hierarchy ID Message - Flit Mode..... | 254 |
| Figure 2-63 | LTR Message - Non-Flit Mode | 256 |
| Figure 2-64 | LTR Message - Flit Mode | 256 |
| Figure 2-65 | OBFF Message - Non-Flit Mode | 257 |
| Figure 2-66 | OBFF Message - Flit Mode | 258 |
| Figure 2-67 | PTM Request/Response Message - Non-Flit Mode..... | 259 |
| Figure 2-68 | PTM ResponseD Message - Non-Flit Mode | 260 |
| Figure 2-69 | PTM Request/Response Message - Flit Mode | 260 |
| Figure 2-70 | PTM ResponseD Message - Flit Mode | 261 |
| Figure 2-71 | IDE Sync Message for Link IDE Stream - Non-Flit Mode..... | 262 |
| Figure 2-72 | IDE Sync Message for Link IDE Stream - Flit Mode | 263 |
| Figure 2-73 | IDE Sync Message for Selective IDE Stream - Non-Flit Mode..... | 263 |
| Figure 2-74 | IDE Sync Message for Selective IDE Stream - Flit Mode | 264 |
| Figure 2-75 | IDE Fail Message for Link IDE Stream - Non-Flit Mode | 265 |
| Figure 2-76 | IDE Fail Message for Link IDE Stream - Flit Mode | 266 |
| Figure 2-77 | IDE Fail Message for Selective IDE Stream - Non-Flit Mode | 266 |
| Figure 2-78 | IDE Fail Message for Selective IDE Stream - Flit Mode | 267 |
| Figure 2-79 | Completion Header Format - Non-Flit Mode | 269 |
| Figure 2-80 | (Non-ARI) Completer ID..... | 270 |
| Figure 2-81 | ARI Completer ID | 270 |
| Figure 2-82 | Completion Header Base Format - Non-UUIO Flit Mode | 272 |
| Figure 2-83 | Completion Header Base Format – UIOWrCpl and UIORdCpl  | 272 |

Base 6.4 vs Base 6.3

| | | |
|-------------|--|-----|
| Figure 2-84 | Completion Header Base Format - UIORdCplD | 273 |
| Figure 2-85 | Flit Mode Local TLP Prefix..... | 276 |
| Figure 2-86 | OHC-E1..... | 279 |
| Figure 2-87 | OHC-E2..... | 279 |
| Figure 2-88 | OHC-E4..... | 280 |
| Figure 2-89 | Flowchart for Handling of Received TLPs | 282 |
| Figure 2-90 | Flowchart for Switch Handling of TLPs..... | 284 |
| Figure 2-91 | Flowchart for Handling of Received | 289 |
| Figure 2-92 | Example Completion Data when some Byte Enables are 0b..... | 292 |
| Figure 2-93 | Deadlock Examples with Intersystem Interconnects | 303 |
| Figure 2-94 | Virtual Channel Concept - An Illustration..... | 310 |
| Figure 2-95 | Virtual Channel Concept - Switch Internals (Upstream Flow)..... | 310 |
| Figure 2-96 | An Example of TC/VC Configurations | 313 |
| Figure 2-97 | Relationship Between Requester and Ultimate Completer | 314 |
| Figure 2-98 | Credit Block Example | 323 |
| Figure 2-99 | Calculation of 32-bit ECRC for TLP End to End Data Integrity Protection | 340 |
| Figure 3-1 | Layering Diagram Highlighting the Data Link Layer | 349 |
| Figure 3-2 | Data Link Control and Management State Machine | 351 |
| Figure 3-3 | VC0 Flow Control Initialization Example with 8b/10b Encoding-based Framing | 363 |
| Figure 3-4 | DLLP Type and CRC Fields (Non-Flit Mode) | 365 |
| Figure 3-5 | DLLP Type Field (Flit Mode) | 365 |
| Figure 3-6 | Data Link Layer Packet Format for Ack and Nak (Non-Flit Mode) | 370 |
| Figure 3-7 | Data Link Layer Packet Format for NOP | 370 |
| Figure 3-8 | Data Link Layer Packet Format for NOP2 (Flit Mode) | 370 |
| Figure 3-9 | Data Link Layer Packet Format for InitFC1..... | 371 |
| Figure 3-10 | Data Link Layer Packet Format for InitFC2..... | 372 |
| Figure 3-11 | Data Link Layer Packet Format for UpdateFC | 373 |
| Figure 3-12 | Data Link Layer Packet Format for Power Management | 373 |
| Figure 3-13 | Data Link Layer Packet Format for Vendor-Specific | 373 |
| Figure 3-14 | Data Link Layer Packet Format for | 374 |
| Figure 3-15 | Data Link Layer Packet Format for | 374 |
| Figure 3-16 | Diagram of CRC Calculation for DLLPs | 376 |
| Figure 3-17 | TLP with LCRC and TLP Sequence Number Applied - Non-Flit Mode | 377 |
| Figure 3-18 | TLP Following Application of TLP Sequence Number and 4 Bits | 379 |
| Figure 3-19 | Calculation of LCRC..... | 381 |
| Figure 3-20 | Received DLLP Error Check Flowchart | 385 |
| Figure 3-21 | Ack/Nak DLLP Processing Flowchart | 386 |
| Figure 3-22 | Receive Data Link Layer Handling of TLPs Flowchart | 389 |
| Figure 4-1 | Layering Diagram Highlighting Physical Layer | 393 |
| Figure 4-2 | Character to Symbol Mapping | 396 |
| Figure 4-3 | Bit Transmission Order on Physical Lanes - x1 Example | 396 |
| Figure 4-4 | Bit Transmission Order on Physical Lanes - x4 Example | 397 |
| Figure 4-5 | TLP with Framing Symbols Applied | 400 |
| Figure 4-6 | DLLP with Framing Symbols Applied | 401 |
| Figure 4-7 | Framed TLP on a x1 Link | 401 |
| Figure 4-8 | Framed TLP on a x2 Link | 402 |
| Figure 4-9 | Framed TLP on a x4 Link | 402 |
| Figure 4-10 | LFSR with 8b/10b Scrambling Polynomial..... | 403 |
| Figure 4-11 | Example of Bit Transmission Order in a x1 Link Showing 130 Bits of a Block | 404 |
| Figure 4-12 | Example of Bit Placement in a x4 Link with One Block per Lane | 405 |
| Figure 4-13 | Layout of Framing Tokens..... | 408 |

Base 6.4 vs Base 6.3

| | | |
|-------------|--|-----|
| Figure 4-14 | TLP and DLLP Layout | 410 |
| Figure 4-15 | Packet Transmission in a x8 Link | 410 |
| Figure 4-16 | Nullified TLP Layout in a x8 Link with Other Packets | 410 |
| Figure 4-17 | SKP Ordered Set of Length 66-bit in a x8 Link | 411 |
| Figure 4-18 | LFSR with Scrambling Polynomial in 8.0 GT/s and Above Data Rate | 418 |
| Figure 4-19 | Alternate Implementation of the LFSR for Descrambling | 420 |
| Figure 4-20 | Precoding working the scrambler/ de-scrambler..... | 422 |
| Figure 4-21 | Example of Symbol placement in a x4 Link with 1b/1b encoding | 425 |
| Figure 4-22 | Transmit side at 64.0 GT/s..... | 426 |
| Figure 4-23 | Receive side at 64.0 GT/s..... | 426 |
| Figure 4-24 | PAM4 Signaling at UI level: Voltage levels, 2-bit encoding, and their corresponding DC balance values..... | 427 |
| Figure 4-25 | The Sequence of Gray Coding, Precoding, and PAM4 voltage translation on an aligned 2-bit boundary on a per Lane | 428 |
| Figure 4-26 | Processing of Ordered Sets during or at the end of a Data Stream in Flit ↓↓mode↓ ↑↑Mode↑ at 64.0 GT/s Data Rate | 436 |
| Figure 4-27 | Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Transmit side..... | 441 |
| Figure 4-28 | Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Receive side..... | 442 |
| Figure 4-29 | DLP Byte to Bit Number Assignment..... | 446 |
| Figure 4-30 | DLP Bit usage..... | 446 |
| Figure 4-31 | Optimized_Update_FC..... | 447 |
| Figure 4-32 | Flit_Marker..... | 448 |
| Figure 4-33 | Flit Ack, Nak, and Discard Rules Flow Chart (Zoom-In to View) | 468 |
| Figure 4-34 | Flit Ack/Nak/Replay Example | 473 |
| Figure 4-35 | ↑↑NOP Flit Common Header↑ | 475 |
| Figure 4-36 | NOP.Empty Flit Payload | 477 |
| Figure 4-37 | NOP.Debug Flit Debug Chunk | 477 |
| Figure 4-38 | Example Debug Chunk with one DW Debug Chunk Heaader and one DW of Debug Chunk Payload .. | 478 |
| Figure 4-39 | Example Debug Chunk with two DW Debug Chunk Heaader and one DW of Debug Chunk Payload .. | 479 |
| Figure 4-40 | Example Debug Chunk with four DW Debug Chunk Heaader and one DW of Debug Chunk Payload | 479 |
| Figure 4-41 | Example NOP.Debug Flit Payload with a single Debug Chunk with a one DW Debug Chunk Header. | 480 |
| Figure 4-42 | Example NOP.Debug Flit Payload with multiple Debug Chunks with one DW Debug Chunk Headers..... | 481 |
| Figure 4-43 | Empty Debug Chunk | 483 |
| Figure 4-44 | Start Capture Trigger Debug Chunk | 483 |
| Figure 4-45 | Stop Capture Trigger Debug Chunk | 484 |
| Figure 4-46 | FC Information Tracked by Transmitter Debug Chunk | 485 |
| Figure 4-47 | ↓↓FC Information Tracked by Reciever Debug Chunk↓ ↑↑FC Information Tracked by Receiver Debug Chunk↑ | 487 |
| Figure 4-48 | Flit Mode Transmitter Retry Flags and Counters Debug Chunk | 489 |
| Figure 4-49 | Flit Mode Receiver Retry Flags and Counters Debug Chunk | 490 |
| Figure 4-50 | Buffer Occupancy Debug Chunk..... | 492 |
| Figure 4-51 | Link Debug Request Debug Chunk | 493 |
| Figure 4-52 | NOP.Vendor Flit Payload | 493 |
| Figure 4-53 | CRC generation/ checking in Flit | 494 |
| Figure 4-54 | FEC Table: i to α^i | 496 |
| Figure 4-55 | FEC Log Table: α^i to i | 497 |
| Figure 4-56 | H-matrix of the FEC | 498 |
| Figure 4-57 | Weight of check bits for different Bytes/bits | 499 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|-----|
| Figure 4-58 | ECC Decoder function | 501 |
| Figure 4-59 | 3-way ECC decode followed by CRC check of flit on the Receive side | 502 |
| Figure 4-60 | 8.0 GT/s Equalization Flow | 513 |
| Figure 4-61 | 16.0 GT/s Equalization Flow | 513 |
| Figure 4-62 | 64.0 GT/s Equalization Flow | 514 |
| Figure 4-63 | Equalization Bypass Example | 515 |
| Figure 4-64 | Alternate Protocol Negotiation and Equalization Bypass LTSSM States | 540 |
| Figure 4-65 | Electrical Idle Exit Ordered Set for 8.0 GT/s to 32.0 GT/s Data Rates (EIEOS) | 545 |
| Figure 4-66 | Example of L0p flow in a x16 Link | 565 |
| Figure 4-67 | Main State Diagram for Link Training and Status State Machine | 569 |
| Figure 4-68 | Detect Substate Machine | 571 |
| Figure 4-69 | Polling Substate Machine | 579 |
| Figure 4-70 | Configuration Substate Machine | 599 |
| Figure 4-71 | Recovery Substate Machine | 631 |
| Figure 4-72 | L0s Substate Machine | 636 |
| Figure 4-73 | L1 Substate Machine | 638 |
| Figure 4-74 | L2 Substate Machine | 639 |
| Figure 4-75 | Loopback Substate Machine | 647 |
| Figure 4-76 | Margin PHY Payload for Control SKP Ordered Set with 1b/1b Encoding | 656 |
| Figure 4-77 | LFSR PHY Payload for Control SKP Ordered Set with 1b/1b Encoding | 656 |
| Figure 4-78 | Polling.Compliance PHY Payload for Control SKP Ordered Set with 1b/1b Encoding | 657 |
| Figure 4-79 | Receiver Number Assignment | 675 |
| Figure 4-80 | Supported Retimer Topologies | 691 |
| Figure 4-81 | ↑↑Supported Retimer Topology Using an Optical Retimer Solution↑↑ | 692 |
| | ECN: Base 6.3 Optical△↔ | |
| Figure 4-82 | ↑↑Optical Aware Retimer Example – Flow Diagram of Steps 1 to 6↑↑ | 696 |
| Figure 4-83 | ↑↑Sample Retimer Topology Using PCIe Optical Aware Retimers↑↑ | 697 |
| Figure 4-84 | ↑↑Flit Interleaving in a x8 Link↑↑ | 699 |
| Figure 4-85 | ↑↑Flit Interleaving of a x8 Link in a 1:2 Electrical Lane to Optical Channel Mapping↑↑ | 700 |
| Figure 4-86 | ↑↑Flit Interleaving of a x8 Link in a 2:1 Electrical Lane to Optical Channel MappingLink↑↑ | 700 |
| Figure 4-87 | Retimer CLKREQ# Connection Topology | 730 |
| Figure 5-1 | Link Power Management State Flow Diagram | 738 |
| Figure 5-2 | Entry into the L1 Link State | 746 |
| Figure 5-3 | Exit from L1 Link State Initiated by Upstream Component | 748 |
| Figure 5-4 | Conceptual Diagrams Showing Two Example Cases of WAKE# Routing | 751 |
| Figure 5-5 | A Conceptual PME Control State Machine | 754 |
| Figure 5-6 | L1 Transition Sequence Ending with a Rejection (L0s Enabled) | 765 |
| Figure 5-7 | L1 Successful Transition Sequence | 766 |
| Figure 5-8 | Example of L1 Exit Latency Computation | 767 |
| Figure 5-9 | State Diagram for L1 PM Substates | 773 |
| Figure 5-10 | Downstream Port with a Single PLL | 774 |
| Figure 5-11 | Multiple Downstream Ports with a shared PLL | 775 |
| Figure 5-12 | Example: L1.1 Waveforms Illustrating Upstream Port Initiated Exit | 777 |
| Figure 5-13 | Example: L1.1 Waveforms Illustrating Downstream Port Initiated Exit | 778 |
| Figure 5-14 | L1.2 Substates | 779 |
| Figure 5-15 | Example: Illustration of Boundary Condition due to Different Sampling of CLKREQ# | 780 |
| Figure 5-16 | Example: L1.2 Waveforms Illustrating Upstream Port Initiated Exit | 781 |
| Figure 5-17 | Example: L1.2 Waveforms Illustrating Downstream Port Initiated Exit | 782 |
| Figure 5-18 | Function Power Management State Transitions | 786 |
| Figure 5-19 | PCI Express Bridge Power Management Diagram | 789 |
| Figure 6-1 | Error Classification | 802 |
| Figure 6-2 | Flowchart Showing Sequence of Device Error Signaling and Logging Operations | 817 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Figure 6-3 | Pseudo Logic Diagram for Selected Error Message Control and Status Bits | 818 |
| Figure 6-4 | TC Filtering Example | 837 |
| Figure 6-5 | TC to VC Mapping Example | 837 |
| Figure 6-6 | An Example of Traffic Flow Illustrating Ingress and Egress | 838 |
| Figure 6-7 | An Example of Differentiated Traffic Flow Through a Switch..... | 839 |
| Figure 6-8 | Switch Arbitration Structure..... | 839 |
| Figure 6-9 | VC ID and Priority Order - An Example..... | 840 |
| Figure 6-10 | Multi-Function Arbitration Model..... | 843 |
| Figure 6-11 | Root Complex Represented as a Single Component..... | 890 |
| Figure 6-12 | Root Complex Represented as Multiple Components..... | 890 |
| Figure 6-13 | Example System Topology with ARI Devices..... | 905 |
| Figure 6-14 | Segmentation of the Multicast Address Range | 906 |
| Figure 6-15 | Latency Fields Format for LTR Messages..... | 921 |
| Figure 6-16 | CLKREQ# and Clock Power Management | 925 |
| Figure 6-17 | Use of LTR and Clock Power Management..... | 926 |
| Figure 6-18 | Codes and Equivalent WAKE# Patterns | 927 |
| Figure 6-19 | Example Platform Topology Showing a Link Where OBFF is Carried by Messages..... | 928 |
| Figure 6-20 | PASID TLP Prefix | 931 |
| Figure 6-21 | Example System Topologies using PTM | 935 |
| Figure 6-22 | Precision Time Measurement Link Protocol | 936 |
| Figure 6-23 | Precision Time Measurement Example | 938 |
| Figure 6-24 | PTM Requester Operation | 941 |
| Figure 6-25 | PTM Timestamp Capture Example | 944 |
| Figure 6-26 | Example Illustrating Application of Enhanced Allocation | 950 |
| Figure 6-27 | Emergency Power Reduction State: Example Add-in Card | 954 |
| Figure 6-28 | FPB High Level Diagram and Example Topology | 959 |
| Figure 6-29 | Example Illustrating “Flattening” of a Switch | 960 |
| Figure 6-30 | Vector Mechanism for Address Range Decoding | 961 |
| Figure 6-31 | Relationship between FPB and non-FPB Decode Mechanisms | 962 |
| Figure 6-32 | Routing IDs (RIDs) and Supported Granularities | 964 |
| Figure 6-33 | Addresses in Memory Below 4 GB and Effect of Granularity..... | 966 |
| Figure 6-34 | VPD Format..... | 971 |
| Figure 6-35 | Example NPEM Configuration using a Downstream Port..... | 976 |
| Figure 6-36 | Example NPEM Configuration using an Upstream Port..... | 977 |
| Figure 6-37 | NPEM Command Flow | 978 |
| Figure 6-38 | Stack Diagram Illustration of Multiple Sessions and Connections | 984 |
| Figure 6-39 | Example Showing Relationships of Software and Hardware Elements | 985 |
| Figure 6-40 | DOE Data Object Format | 985 |
| Figure 6-41 | DOE Data Object Header 1 | 986 |
| Figure 6-42 | DOE Data Object Header 2 | 986 |
| Figure 6-43 | DOE Discovery Request Data Object Contents (3rd DW) | 987 |
| Figure 6-44 | DOE Discovery Response Data Object Contents (3rd DW)..... | 987 |
| Figure 6-45 | DOE Discovery Response Data Object Contents (4th DW)..... | 988 |
| Figure 6-46 | CMA-SPDM as Part of a Layered Architecture | 994 |
| Figure 6-47 | Example System Showing Multiple Access Mechanisms..... | 996 |
| Figure 6-48 | Example Add-In-Card Supporting CMA-SPDM | 997 |
| Figure 6-49 | Byte Mapping of SPDM Messages Including Example Payload | 1000 |
| Figure 6-50 | Byte Mapping of Secured CMA-SPDM Messages Including Example Payload | 1001 |
| Figure 6-51 | Example DMWr Data Payload Template | 1006 |
| Figure 6-52 | IDE Secures TLPs Between Ports | 1008 |
| Figure 6-53 | IDE Stream State Machine..... | 1012 |
| Figure 6-54 | IDE Stream State Machine..... | 1013 |

Base 6.4 vs Base 6.3

| | | |
|--------------|--|------|
| Figure 6-55 | IDE Key Management (IDE_KM) and Related Specifications & Capabilities | 1015 |
| Figure 6-56 | Query (QUERY) Data Object | 1020 |
| Figure 6-57 | Query Response (QUERY_RESP) Data Object | 1021 |
| Figure 6-58 | Key Programming (KEY_PROG) Data Object with Example 256b Key | 1021 |
| Figure 6-59 | Key Programming Acknowledgement (KP_ACK) Data Object | 1022 |
| Figure 6-60 | Key Set Go (K_SET_GO) Data Object | 1022 |
| Figure 6-61 | Key Set Stop (K_SET_STOP) Data Object | 1022 |
| Figure 6-62 | Key Set Go/Stop Acknowledgement (K_GOSTOP_ACK) Data Object | 1022 |
| Figure 6-63 | IDE_KM Example | 1024 |
| Figure 6-64 | IDE TLP Prefix (NFM)..... | 1025 |
| Figure 6-65 | MAC Layout..... | 1026 |
| Figure 6-66 | Example of IDE TLP for a Link IDE Stream without Aggregation (Non-Flit Mode) | 1027 |
| Figure 6-67 | IDE TLP – Example Showing Aggregation of Two TLPs for a Link IDE Stream (Non-Flit Mode) | 1027 |
| Figure 6-68 | IDE TLP – Example of IDE TLP for a Selective IDE Stream without Aggregation (Non-Flit Mode)..... | 1027 |
| Figure 6-69 | IDE TLP – Example Showing Aggregation of Two TLPs for a Selective IDE Stream (Non-Flit Mode) . | 1028 |
| Figure 6-70 | Example of IDE TLP for a Link IDE Stream without Aggregation (Flit Mode) | 1028 |
| Figure 6-71 | IDE TLP – Example Showing Aggregation of Two TLPs for a Link IDE Stream (Flit Mode) | 1028 |
| Figure 6-72 | IDE TLP – Example of IDE TLP for a Selective IDE Stream without Aggregation (Flit Mode) | 1029 |
| Figure 6-73 | IDE TLP – Example Showing Aggregation of Two TLPs for a Selective IDE Stream (Flit Mode) | 1029 |
| Figure 6-74 | High Level Flow For Partial Header Encryption | 1030 |
| Figure 6-75 | Partial Header Encryption in NFM with Byte Enables..... | 1032 |
| Figure 6-76 | Partial Header Encryption in NFM without Byte Enables..... | 1033 |
| Figure 6-77 | Partial Header Encryption in FM with OHC-A1..... | 1034 |
| Figure 6-78 | Partial Header Encryption in FM without OHC-A1 | 1035 |
| Figure 6-79 | Example Illustrating PCRC Application to Two Aggregated IDE TLPs for a Link IDE Stream (NFM) ... | 1036 |
| Figure 6-80 | Example – Posted Requests Allowed to Bypass Non-Posted Requests | 1045 |
| Figure 6-81 | Example – Non-Posted Requests Never Allowed to Bypass Posted Requests..... | 1046 |
| Figure 6-82 | Example – Secure Non-Posted Request Reordering Not Allowed Over PCIe Fabric | 1046 |
| Figure 6-83 | MMIO Register Blocks..... | 1054 |
| Figure 6-84 | MCAP Register Block | 1055 |
| Figure 6-85 | MCAP Array Register Block..... | 1056 |
| Figure 6-86 | MCAP Array Register 1 | 1056 |
| Figure 6-87 | MCAP Array Register 2 | 1057 |
| Figure 6-88 | MCAP Header Register Block | 1057 |
| Figure 6-89 | MCAP Header Register 1 | 1057 |
| Figure 6-90 | MCAP Header Register 2 | 1058 |
| Figure 6-91 | MCAP Header Register 3 | 1058 |
| Figure 6-92 | MCAP Header Register 4 | 1059 |
| Figure 6-93 | MMB Registers | 1061 |
| Figure 6-94 | MMB Capabilities Register | 1062 |
| Figure 6-95 | MMB Control Register..... | 1063 |
| Figure 6-96 | MMB Command Register..... | 1064 |
| Figure 6-97 | MMB Status Register..... | 1065 |
| Figure 6-98 | MMB Payload Registers | 1067 |
| Figure 6-99 | MMPT Registers | 1067 |
| Figure 6-100 | MMPT Capabilities Register | 1068 |
| Figure 6-101 | MMPT Control Register..... | 1068 |
| Figure 6-102 | MMPT Receive Message Notification Register | 1069 |
| Figure 6-103 | MDVS Register Block | 1070 |
| Figure 6-104 | MDVS Register Block Header Register 1 | 1070 |
| Figure 6-105 | MDVS Register Block Header Register 2 | 1071 |
| Figure 6-106 | MDVS Register Block Header Register 3 | 1071 |

Base 6.4 vs Base 6.3

| | | |
|-------------|--|------|
| Figure 7-1 | PCI Express Root Complex Device Mapping..... | 1079 |
| Figure 7-2 | PCI Express Switch Device Mapping | 1080 |
| Figure 7-3 | PCI Express Configuration Space Layout | 1081 |
| Figure 7-4 | Common Configuration Space Header | 1092 |
| Figure 7-5 | Command Register..... | 1093 |
| Figure 7-6 | Status Register..... | 1096 |
| Figure 7-7 | Class Code Register | 1099 |
| Figure 7-8 | Header Type Register | 1100 |
| Figure 7-9 | BIST Register..... | 1101 |
| Figure 7-10 | Type 0 Configuration Space Header | 1103 |
| Figure 7-11 | Base Address Register for Memory | 1104 |
| Figure 7-12 | Base Address Register for I/O..... | 1104 |
| Figure 7-13 | Expansion ROM Base Address Register | 1109 |
| Figure 7-14 | Type 1 Configuration Space Header | 1112 |
| Figure 7-15 | Secondary Status Register | 1115 |
| Figure 7-16 | Bridge Control Register | 1118 |
| Figure 7-17 | PCI Power Management Capability Structure | 1120 |
| Figure 7-18 | Power Management Capabilities Register | 1121 |
| Figure 7-19 | Power Management Control/Status Register..... | 1123 |
| Figure 7-20 | Power Management Data Register | 1125 |
| Figure 7-21 | PCI Express Capability Structure | 1127 |
| Figure 7-22 | PCI Express Capability List Register..... | 1128 |
| Figure 7-23 | PCI Express Capabilities Register..... | 1128 |
| Figure 7-24 | Device Capabilities Register..... | 1130 |
| Figure 7-25 | Device Control Register | 1134 |
| Figure 7-26 | Device Status Register..... | 1141 |
| Figure 7-27 | Link Capabilities Register..... | 1143 |
| Figure 7-28 | Link Control Register..... | 1147 |
| Figure 7-29 | Link Status Register..... | 1154 |
| Figure 7-30 | Slot Capabilities Register | 1156 |
| Figure 7-31 | Slot Control Register | 1159 |
| Figure 7-32 | Slot Status Register | 1162 |
| Figure 7-33 | Root Control Register | 1164 |
| Figure 7-34 | Root Capabilities Register | 1165 |
| Figure 7-35 | Root Status Register | 1166 |
| Figure 7-36 | Device Capabilities 2 Register | 1167 |
| Figure 7-37 | Device Control 2 Register | 1172 |
| Figure 7-38 | Link Capabilities 2 Register..... | 1176 |
| Figure 7-39 | Link Control 2 Register | 1179 |
| Figure 7-40 | Link Status 2 Register | 1182 |
| Figure 7-41 | Slot Capabilities 2 Register | 1186 |
| Figure 7-42 | PCI Express Extended Configuration Space Layout | 1187 |
| Figure 7-43 | PCI Express Extended Capability Header | 1188 |
| Figure 7-44 | MSI Capability Structure for 32-bit Message Address..... | 1189 |
| Figure 7-45 | MSI Capability Structure for 64-bit Message Address..... | 1189 |
| Figure 7-46 | MSI Capability Structure for 32-bit Message Address and PVM..... | 1190 |
| Figure 7-47 | MSI Capability Structure for 64-bit Message Address and PVM..... | 1190 |
| Figure 7-48 | MSI Capability Header..... | 1191 |
| Figure 7-49 | Message Control Register for MSI | 1192 |
| Figure 7-50 | Message Address Register for MSI | 1193 |
| Figure 7-51 | Message Upper Address Register for MSI | 1194 |
| Figure 7-52 | Message Data Register for MSI | 1194 |

| | | |
|--------------|---|------|
| Figure 7-53 | Extended Message Data Register for MSI..... | 1195 |
| Figure 7-54 | Mask Bits Register for MSI | 1196 |
| Figure 7-55 | Pending Bits Register for MSI..... | 1196 |
| Figure 7-56 | MSI-X Capability Structure..... | 1197 |
| Figure 7-57 | MSI-X Table Structure..... | 1198 |
| Figure 7-58 | MSI-X PBA Structure | 1198 |
| Figure 7-59 | MSI-X Capability Header | 1200 |
| Figure 7-60 | Message Control Register for MSI-X..... | 1201 |
| Figure 7-61 | Table Offset/Table BIR Register for MSI-X..... | 1202 |
| Figure 7-62 | PBA Offset/PBA BIR Register for MSI-X | 1202 |
| Figure 7-63 | Message Address Register for MSI-X Table Entries..... | 1203 |
| Figure 7-64 | Message Upper Address Register for MSI-X Table Entries | 1204 |
| Figure 7-65 | Message Data Register for MSI-X Table Entries | 1204 |
| Figure 7-66 | Vector Control Register for MSI-X Table Entries | 1205 |
| Figure 7-67 | Pending Bits Register for MSI-X PBA Entries | 1205 |
| Figure 7-68 | Secondary PCI Express Extended Capability Structure | 1207 |
| Figure 7-69 | Secondary PCI Express Extended Capability Header | 1208 |
| Figure 7-70 | Link Control 3 Register..... | 1208 |
| Figure 7-71 | Lane Error Status Register | 1209 |
| Figure 7-72 | Lane Equalization Control Register | 1210 |
| Figure 7-73 | Lane Equalization Control Register Entry | 1210 |
| Figure 7-74 | Data Link Feature Extended Capability | 1213 |
| Figure 7-75 | Data Link Feature Extended Capability Header | 1213 |
| Figure 7-76 | Data Link Feature Capabilities Register | 1214 |
| Figure 7-77 | Data Link Feature Status Register..... | 1215 |
| Figure 7-78 | Physical Layer 16.0 GT/s Extended Capability..... | 1217 |
| Figure 7-79 | Physical Layer 16.0 GT/s Extended Capability Header..... | 1218 |
| Figure 7-80 | 16.0 GT/s Capabilities Register..... | 1218 |
| Figure 7-81 | 16.0 GT/s Control Register | 1219 |
| Figure 7-82 | 16.0 GT/s Status Register | 1219 |
| Figure 7-83 | 16.0 GT/s Local Data Parity Mismatch Status Register | 1220 |
| Figure 7-84 | 16.0 GT/s First Retimer Data Parity Mismatch Status Register | 1221 |
| Figure 7-85 | 16.0 GT/s Second Retimer Data Parity Mismatch Status Register | 1221 |
| Figure 7-86 | 16.0 GT/s Lane Equalization Control Register Entry | 1222 |
| Figure 7-87 | Physical Layer 32.0 GT/s Extended Capability..... | 1224 |
| Figure 7-88 | Physical Layer 32.0 GT/s Extended Capability Header..... | 1225 |
| Figure 7-89 | 32.0 GT/s Capabilities Register..... | 1225 |
| Figure 7-90 | 32.0 GT/s Control Register | 1226 |
| Figure 7-91 | 32.0 GT/s Status Register | 1227 |
| Figure 7-92 | Received Modified TS Data 1 Register | 1228 |
| Figure 7-93 | Received Modified TS Data 2 Register | 1229 |
| Figure 7-94 | Transmitted Modified TS Data 1 Register | 1230 |
| Figure 7-95 | Transmitted Modified TS Data 2 Register | 1231 |
| Figure 7-96 | 32.0 GT/s Lane Equalization Control Register Entry | 1233 |
| Figure 7-97 | Physical Layer 64.0 GT/s Extended Capability..... | 1234 |
| Figure 7-98 | Physical Layer 64.0 GT/s Extended Capability Header..... | 1235 |
| Figure 7-99 | 64.0 GT/s Capabilities Register..... | 1235 |
| Figure 7-100 | 64.0 GT/s Control Register | 1236 |
| Figure 7-101 | 64.0 GT/s Status Register | 1236 |
| Figure 7-102 | 64.0 GT/s Lane Equalization Control Register Entry | 1238 |
| Figure 7-103 | Flit Logging Extended Capability Structure | 1239 |
| Figure 7-104 | Flit Logging Extended Capability Header..... | 1240 |

Base 6.4 vs Base 6.3

| | | |
|--------------|---|------|
| Figure 7-105 | Flit Error Log 1 Register | 1241 |
| Figure 7-106 | Flit Error Log 2 Register | 1243 |
| Figure 7-107 | Flit Error Counter Control Register | 1244 |
| Figure 7-108 | Flit Error Counter Status Register | 1245 |
| Figure 7-109 | FBER Measurement Control Register | 1246 |
| Figure 7-110 | FBER Measurement Status 1 Register..... | 1247 |
| Figure 7-111 | FBER Measurement Status 2 Register..... | 1247 |
| Figure 7-112 | FBER Measurement Status 3 Register..... | 1248 |
| Figure 7-113 | FBER Measurement Status 4 Register..... | 1248 |
| Figure 7-114 | FBER Measurement Status 5 Register..... | 1249 |
| Figure 7-115 | FBER Measurement Status 6 Register..... | 1249 |
| Figure 7-116 | FBER Measurement Status 7 Register..... | 1249 |
| Figure 7-117 | FBER Measurement Status 8 Register..... | 1250 |
| Figure 7-118 | FBER Measurement Status 9 Register..... | 1250 |
| Figure 7-119 | FBER Measurement Status 10 Register..... | 1251 |
| Figure 7-120 | Device 3 Extended Capability Structure | 1251 |
| Figure 7-121 | Device 3 Extended Capability Header | 1252 |
| Figure 7-122 | Device Capabilities 3 Register..... | 1252 |
| Figure 7-123 | Device Control 3 Register | 1255 |
| Figure 7-124 | Device Status 3 Register | 1258 |
| Figure 7-125 | Lane Margining at the Receiver Extended Capability | 1260 |
| Figure 7-126 | Lane Margining at the Receiver Extended Capability Header | 1261 |
| Figure 7-127 | MARGINING Port Capabilities Register | 1261 |
| Figure 7-128 | MARGINING Port Status Register | 1262 |
| Figure 7-129 | Lane N: Margining Control Register Entry | 1263 |
| Figure 7-130 | Lane N: Margining Lane Status Register Entry | 1264 |
| Figure 7-131 | ACS Extended Capability | 1265 |
| Figure 7-132 | ACS Extended Capability Header | 1265 |
| Figure 7-133 | ACS Capability Register | 1266 |
| Figure 7-134 | ACS Control Register | 1267 |
| Figure 7-135 | Egress Control Vector Register..... | 1270 |
| Figure 7-136 | Power Budgeting Extended Capability..... | 1271 |
| Figure 7-137 | Power Budgeting Extended Capability Header..... | 1272 |
| Figure 7-138 | Power Budgeting Control Register | 1273 |
| Figure 7-139 | Power Budgeting Data Register | 1275 |
| Figure 7-140 | Power Budgeting Capability Register..... | 1279 |
| Figure 7-141 | Power Budgeting Sense Detect Register | 1281 |
| Figure 7-142 | LTR Extended Capability Structure | 1284 |
| Figure 7-143 | LTR Extended Capability Header | 1284 |
| Figure 7-144 | Max Snoop Latency Register..... | 1285 |
| Figure 7-145 | Max No-Snoop Latency Register..... | 1285 |
| Figure 7-146 | ECN: Base 6.3 LTR-MFDLTR Capabilities Register | 1286 |
| Figure 7-147 | L1 PM Substates Extended Capability | 1287 |
| Figure 7-148 | L1 PM Substates Extended Capability Header | 1287 |
| Figure 7-149 | L1 PM Substates Capabilities Register..... | 1288 |
| Figure 7-150 | L1 PM Substates Control 1 Register | 1289 |
| Figure 7-151 | L1 PM Substates Control 2 Register | 1291 |
| Figure 7-152 | L1 PM Substates Status Register..... | 1292 |
| Figure 7-153 | Advanced Error Reporting Extended Capability - Functions that do not support Flit Mode Structure..... | 1293 |
| Figure 7-154 | Advanced Error Reporting Extended Capability - Functions that support Flit Mode Structure | 1294 |
| Figure 7-155 | Advanced Error Reporting Extended Capability Header | 1295 |

Base 6.4 vs Base 6.3

| | | |
|--------------|--|------|
| Figure 7-156 | Uncorrectable Error Status Register | 1296 |
| Figure 7-157 | Uncorrectable Error Mask Register | 1298 |
| Figure 7-158 | Uncorrectable Error Severity Register | 1301 |
| Figure 7-159 | Correctable Error Status Register | 1303 |
| Figure 7-160 | Correctable Error Mask Register | 1304 |
| Figure 7-161 | Advanced Error Capabilities and Control Register | 1306 |
| Figure 7-162 | Header Log Register | 1308 |
| Figure 7-163 | Root Error Command Register | 1309 |
| Figure 7-164 | Root Error Status Register | 1311 |
| Figure 7-165 | Error Source Identification Register | 1313 |
| Figure 7-166 | TLP Prefix Log Register | 1314 |
| Figure 7-167 | First DW of Enhanced Allocation Capability | 1315 |
| Figure 7-168 | Second DW of Enhanced Allocation Capability | 1316 |
| Figure 7-169 | First DW of Each Entry for Enhanced Allocation Capability | 1316 |
| Figure 7-170 | Format of Entry for Enhanced Allocation Capability | 1318 |
| Figure 7-171 | Example Entry with 64b Base and 64b MaxOffset | 1320 |
| Figure 7-172 | Example Entry with 64b Base and 32b MaxOffset | 1320 |
| Figure 7-173 | Example Entry with 32b Base and 64b MaxOffset | 1321 |
| Figure 7-174 | Example Entry with 32b Base and 32b MaxOffset | 1321 |
| Figure 7-175 | Resizable BAR Extended Capability | 1323 |
| Figure 7-176 | Resizable BAR Extended Capability Header | 1323 |
| Figure 7-177 | Resizable BAR Capability Register | 1324 |
| Figure 7-178 | Resizable BAR Control Register | 1326 |
| Figure 7-179 | VF Resizable BAR Extended Capability | 1329 |
| Figure 7-180 | VF Resizable BAR Extended Capability Header | 1330 |
| Figure 7-181 | VF Resizable BAR Control Register | 1331 |
| Figure 7-182 | ARI Extended Capability | 1332 |
| Figure 7-183 | ARI Extended Capability Header | 1332 |
| Figure 7-184 | ARI Capability Register | 1333 |
| Figure 7-185 | ARI Control Register | 1334 |
| Figure 7-186 | PASID Extended Capability Structure | 1335 |
| Figure 7-187 | PASID Extended Capability Header | 1335 |
| Figure 7-188 | PASID Capability Register | 1336 |
| Figure 7-189 | PASID Control Register | 1337 |
| Figure 7-190 | FRS Queueing Extended Capability | 1338 |
| Figure 7-191 | FRS Queueing Extended Capability Header | 1338 |
| Figure 7-192 | FRS Queueing Capability Register | 1339 |
| Figure 7-193 | FRS Queueing Status Register | 1340 |
| Figure 7-194 | FRS Queueing Control Register | 1340 |
| Figure 7-195 | FRS Message Queue Register | 1341 |
| Figure 7-196 | FPB Capability Structure | 1342 |
| Figure 7-197 | FPB Capability Header | 1342 |
| Figure 7-198 | FPB Capabilities Register | 1343 |
| Figure 7-199 | FPB RID Vector Control 1 Register | 1345 |
| Figure 7-200 | FPB RID Vector Control 2 Register | 1346 |
| Figure 7-201 | FPB MEM Low Vector Control Register | 1347 |
| Figure 7-202 | FPB MEM High Vector Control 1 Register | 1348 |
| Figure 7-203 | FPB MEM High Vector Control 2 Register | 1350 |
| Figure 7-204 | FPB Vector Access Control Register | 1351 |
| Figure 7-205 | FPB Vector Access Data Register | 1352 |
| Figure 7-206 | Flit Performance Measurement Extended Capability Structure | 1353 |
| Figure 7-207 | Flit Performance Measurement Extended Capability Header | 1353 |

Base 6.4 vs Base 6.3

| | | |
|--------------|--|------|
| Figure 7-208 | Flit Performance Measurement Capability Register | 1354 |
| Figure 7-209 | Flit Performance Measurement Control Register | 1355 |
| Figure 7-210 | Flit Performance Measurement Status Register | 1357 |
| Figure 7-211 | LTSSM Performance Measurement Status Register | 1358 |
| Figure 7-212 | Flit Error Injection Extended Capability Structure..... | 1360 |
| Figure 7-213 | Flit Error Injection Extended Capability Header | 1360 |
| Figure 7-214 | Flit Error Injection Capability Register | 1361 |
| Figure 7-215 | Flit Error Injection Control 1 Register..... | 1361 |
| Figure 7-216 | Flit Error Injection Control 2 Register..... | 1363 |
| Figure 7-217 | Flit Error Injection Status Register | 1364 |
| Figure 7-218 | Ordered Set Error Injection Control 1 Register | 1365 |
| Figure 7-219 | Ordered Set Error Injection Control 2 Register | 1366 |
| Figure 7-220 | Ordered Set Tx Error Injection Status Register | 1367 |
| Figure 7-221 | Ordered Set Rx Error Injection Status Register | 1368 |
| Figure 7-222 | NOP Flit Extended Capability..... | 1370 |
| Figure 7-223 | NOP Flit Extended Capability Header..... | 1370 |
| Figure 7-224 | NOP Flit EditorialCapabilites EditorialCapabilities Register | 1371 |
| Figure 7-225 | NOP Flit Control 1 Register | 1371 |
| Figure 7-226 | NOP Flit Control 2 Register | 1373 |
| Figure 7-227 | NOP Flit Status Register | 1374 |
| Figure 7-228 | Virtual Channel Extended Capability Structure | 1376 |
| Figure 7-229 | Virtual Channel Extended Capability Header..... | 1376 |
| Figure 7-230 | Port VC Capability Register 1 | 1377 |
| Figure 7-231 | Port VC Capability Register 2 | 1378 |
| Figure 7-232 | Port VC Control Register..... | 1379 |
| Figure 7-233 | Port VC Status Register..... | 1380 |
| Figure 7-234 | VC Resource Capability Register | 1380 |
| Figure 7-235 | VC Resource Control Register | 1382 |
| Figure 7-236 | VC Resource Status Register | 1384 |
| Figure 7-237 | Example VC Arbitration Table with 32 Phases..... | 1385 |
| Figure 7-238 | Example Port Arbitration Table with 128 Phases and 2-bit Table Entries | 1386 |
| Figure 7-239 | MFVC Capability Structure | 1387 |
| Figure 7-240 | MFVC Extended Capability Header | 1388 |
| Figure 7-241 | MFVC Port VC Capability Register 1 | 1388 |
| Figure 7-242 | MFVC Port VC Capability Register 2 | 1389 |
| Figure 7-243 | MFVC Port VC Control Register..... | 1390 |
| Figure 7-244 | MFVC Port VC Status Register..... | 1391 |
| Figure 7-245 | MFVC VC Resource Capability Register | 1392 |
| Figure 7-246 | MFVC VC Resource Control Register | 1393 |
| Figure 7-247 | MFVC VC Resource Status Register | 1395 |
| Figure 7-248 | Device Serial Number Extended Capability Structure | 1398 |
| Figure 7-249 | Device Serial Number Extended Capability Header | 1398 |
| Figure 7-250 | Serial Number Register | 1399 |
| Figure 7-251 | Vendor-Specific Capability | 1400 |
| Figure 7-252 | VSEC Capability Structure..... | 1401 |
| Figure 7-253 | Vendor-Specific Extended Capability Header | 1401 |
| Figure 7-254 | Vendor-Specific Header | 1402 |
| Figure 7-255 | Designated Vendor-Specific Extended Capability | 1403 |
| Figure 7-256 | Designated Vendor-Specific Extended Capability Header | 1403 |
| Figure 7-257 | Designated Vendor-Specific Header 1..... | 1404 |
| Figure 7-258 | Designated Vendor-Specific Header 2..... | 1404 |
| Figure 7-259 | RCRB Header Extended Capability Structure..... | 1405 |

Base 6.4 vs Base 6.3

| | | |
|--------------|--|------|
| Figure 7-260 | RCRB Header Extended Capability Header | 1405 |
| Figure 7-261 | RCRB Vendor ID and Device ID register | 1406 |
| Figure 7-262 | RCRB Capabilities register | 1407 |
| Figure 7-263 | RCRB Control register | 1407 |
| Figure 7-264 | Root Complex Link Declaration Extended Capability..... | 1409 |
| Figure 7-265 | Root Complex Link Declaration Extended Capability Header..... | 1409 |
| Figure 7-266 | Element Self Description Register | 1410 |
| Figure 7-267 | Link Entry..... | 1411 |
| Figure 7-268 | Link Description Register | 1411 |
| Figure 7-269 | Link Address for Link Type 0 | 1412 |
| Figure 7-270 | Link Address for Link Type 1 | 1413 |
| Figure 7-271 | Root Complex Internal Link Control Extended Capability..... | 1414 |
| Figure 7-272 | Root Complex Internal Link Control Extended Capability Header..... | 1414 |
| Figure 7-273 | Root Complex Link Capabilities Register | 1415 |
| Figure 7-274 | Root Complex Link Control Register | 1417 |
| Figure 7-275 | Root Complex Link Status Register | 1419 |
| Figure 7-276 | Root Complex Event Collector Endpoint Association Extended Capability | 1420 |
| Figure 7-277 | Root Complex Event Collector Endpoint Association Extended Capability Header | 1420 |
| Figure 7-278 | RCEC Associated Bus Numbers Register | 1421 |
| Figure 7-279 | Multicast Extended Capability Structure..... | 1423 |
| Figure 7-280 | Multicast Extended Capability Header | 1423 |
| Figure 7-281 | Multicast Capability Register | 1424 |
| Figure 7-282 | Multicast Control Register..... | 1425 |
| Figure 7-283 | MC_Base_Address Register..... | 1425 |
| Figure 7-284 | MC_Receive Register | 1426 |
| Figure 7-285 | MC_Block_All Register | 1426 |
| Figure 7-286 | MC_Block_Untranslated Register | 1427 |
| Figure 7-287 | MC_Overlay_BAR Register | 1428 |
| Figure 7-288 | Dynamic Power Allocation Extended Capability Structure | 1428 |
| Figure 7-289 | DPA Extended Capability Header..... | 1429 |
| Figure 7-290 | DPA Capability Register..... | 1429 |
| Figure 7-291 | DPA Latency Indicator Register..... | 1430 |
| Figure 7-292 | DPA Status Register | 1431 |
| Figure 7-293 | DPA Control Register | 1431 |
| Figure 7-294 | DPA Power Allocation Array | 1432 |
| Figure 7-295 | Substate Power Allocation Register (0 to Substate_Max)..... | 1432 |
| Figure 7-296 | TPH Extended Capability Structure..... | 1433 |
| Figure 7-297 | TPH Requester Extended Capability Header..... | 1433 |
| Figure 7-298 | TPH Requester Capability Register..... | 1433 |
| Figure 7-299 | TPH Requester Control Register | 1434 |
| Figure 7-300 | TPH ST Table..... | 1435 |
| Figure 7-301 | TPH ST Table Entry..... | 1436 |
| Figure 7-302 | DPC Extended Capability – Non-Flit Mode | 1437 |
| Figure 7-303 | DPC Extended Capability – Flit Mode | 1438 |
| Figure 7-304 | DPC Extended Capability Header | 1439 |
| Figure 7-305 | DPC Capability Register | 1439 |
| Figure 7-306 | DPC Control Register | 1441 |
| Figure 7-307 | DPC Status Register | 1443 |
| Figure 7-308 | DPC Error Source ID Register | 1444 |
| Figure 7-309 | RP PIO Status Register..... | 1445 |
| Figure 7-310 | RP PIO Mask Register | 1446 |
| Figure 7-311 | RP PIO Severity Register | 1447 |

Base 6.4 vs Base 6.3

| | | |
|--------------|--|------|
| Figure 7-312 | RP PIO SysError Register | 1448 |
| Figure 7-313 | RP PIO Exception Register | 1449 |
| Figure 7-314 | RP PIO Header Log Register | 1450 |
| Figure 7-315 | RP PIO ImpSpec Log Register | 1450 |
| Figure 7-316 | RP PIO TLP Prefix Log Register | 1451 |
| Figure 7-317 | PTM Extended Capability Structure | 1452 |
| Figure 7-318 | PTM Extended Capability Header | 1452 |
| Figure 7-319 | PTM Capability Register | 1453 |
| Figure 7-320 | PTM Control Register | 1454 |
| Figure 7-321 | Readiness Time Reporting Extended Capability | 1456 |
| Figure 7-322 | Readiness Time Encoding | 1456 |
| Figure 7-323 | Readiness Time Reporting Extended Capability Header | 1457 |
| Figure 7-324 | Readiness Time Reporting 1 Register | 1457 |
| Figure 7-325 | Readiness Time Reporting 2 Register | 1458 |
| Figure 7-326 | Hierarchy ID Extended Capability | 1460 |
| Figure 7-327 | Hierarchy ID Extended Capability Header | 1460 |
| Figure 7-328 | Hierarchy ID Status Register | 1461 |
| Figure 7-329 | Hierarchy ID Data Register | 1462 |
| Figure 7-330 | Hierarchy ID GUID 1 Register | 1463 |
| Figure 7-331 | Hierarchy ID GUID 2 Register | 1463 |
| Figure 7-332 | Hierarchy ID GUID 3 Register | 1464 |
| Figure 7-333 | Hierarchy ID GUID 4 Register | 1464 |
| Figure 7-334 | Hierarchy ID GUID 5 Register | 1465 |
| Figure 7-335 | VPD Capability Structure | 1466 |
| Figure 7-336 | VPD Address Register | 1467 |
| Figure 7-337 | VPD Data Register | 1467 |
| Figure 7-338 | NPEM Extended Capability | 1468 |
| Figure 7-339 | NPEM Extended Capability Header | 1468 |
| Figure 7-340 | NPEM Capability Register | 1469 |
| Figure 7-341 | NPEM Control Register | 1470 |
| Figure 7-342 | NPEM Status Register | 1472 |
| Figure 7-343 | Alternate Protocol Extended Capability | 1473 |
| Figure 7-344 | Alternate Protocol Extended Capability Header | 1473 |
| Figure 7-345 | Alternate Protocol Capabilities Register | 1474 |
| Figure 7-346 | Alternate Protocol Control Register | 1474 |
| Figure 7-347 | Alternate Protocol Data 1 Register | 1475 |
| Figure 7-348 | Alternate Protocol Data 2 Register | 1476 |
| Figure 7-349 | Alternate Protocol Selective Enable Mask Register | 1476 |
| Figure 7-350 | Conventional PCI Advanced Features Capability (AF) | 1477 |
| Figure 7-351 | Advanced Features Capability Header | 1477 |
| Figure 7-352 | AF Capabilities Register | 1478 |
| Figure 7-353 | Conventional PCI Advanced Features Control Register | 1478 |
| Figure 7-354 | AF Status Register | 1479 |
| Figure 7-355 | SFI Extended Capability | 1480 |
| Figure 7-356 | SFI Extended Capability Header | 1480 |
| Figure 7-357 | SFI Capability Register | 1481 |
| Figure 7-358 | SFI Control Register | 1482 |
| Figure 7-359 | SFI Status Register | 1485 |
| Figure 7-360 | SFI CAM Address Register | 1486 |
| Figure 7-361 | SFI CAM Data Register | 1486 |
| Figure 7-362 | Subsystem ID and Subsystem Vendor ID Capability | 1487 |
| Figure 7-363 | Subsystem ID and Subsystem Vendor ID Capability Header | 1488 |

| | | |
|--------------|--|------|
| Figure 7-364 | Subsystem ID and Subsystem Vendor ID Capability Data | 1488 |
| Figure 7-365 | Data Object Exchange Extended Capability | 1489 |
| Figure 7-366 | DOE Extended Capability Header | 1489 |
| Figure 7-367 | DOE Capabilities Register | 1490 |
| Figure 7-368 | DOE Control Register..... | 1491 |
| Figure 7-369 | DOE Status Register..... | 1492 |
| Figure 7-370 | DOE Write Data Mailbox Register | 1493 |
| Figure 7-371 | DOE Read Data Mailbox Register | 1493 |
| Figure 7-372 | Shadow Functions Extended Capability Structure | 1496 |
| Figure 7-373 | EditorialShadow Functions Extended Capability Header EditorialShadow Functions Extended Capability Header..... | 1496 |
| Figure 7-374 | EditorialShadow Functions Capability Register EditorialShadow Functions Capability Register | 1497 |
| Figure 7-375 | EditorialShadow Functions Control Register EditorialShadow Functions Control Register | 1497 |
| Figure 7-376 | EditorialShadow Functions Instance Register Entry EditorialShadow Functions Instance Register Entry..... | 1498 |
| Figure 7-377 | IDE Extended Capability Structure | 1499 |
| Figure 7-378 | IDE Extended Capability Header..... | 1499 |
| Figure 7-379 | IDE Capability Register..... | 1500 |
| Figure 7-380 | IDE Control Register | 1502 |
| Figure 7-381 | Link IDE Stream Control Register | 1502 |
| Figure 7-382 | Link IDE Stream Status Register | 1505 |
| Figure 7-383 | Selective IDE Stream Capability Register | 1505 |
| Figure 7-384 | Selective IDE Stream Control Register | 1506 |
| Figure 7-385 | Selective IDE Stream Status Register | 1509 |
| Figure 7-386 | IDE RID Association Register 1 (Offset +00h) | 1510 |
| Figure 7-387 | IDE RID Association Register 2 (Offset +04h) | 1510 |
| Figure 7-388 | IDE Address Association Register 1 (Offset +00h) | 1511 |
| Figure 7-389 | IDE Address Association Register 2 (Offset +04h) | 1511 |
| Figure 7-390 | IDE Address Association Register 3 (Offset +04h) | 1512 |
| Figure 7-391 | Null Capability | 1512 |
| Figure 7-392 | Null Extended Capability | 1513 |
| Figure 7-393 | Streamlined Virtual Channel Extended Capability Structure | 1514 |
| Figure 7-394 | Streamlined Virtual Channel Extended Capability Header | 1514 |
| Figure 7-395 | SVC Port Capability Register 1 | 1515 |
| Figure 7-396 | SVC Port Control Register | 1515 |
| Figure 7-397 | SVC Port Status Register..... | 1516 |
| Figure 7-398 | SVC Resource Capability Register..... | 1517 |
| Figure 7-399 | SVC Resource Control Register | 1517 |
| Figure 7-400 | SVC Resource Status Register | 1519 |
| Figure 7-401 | MRBL Extended Capability..... | 1520 |
| Figure 7-402 | MRBL Extended Capability Header | 1521 |
| Figure 7-403 | MRBL Capabilities Register | 1521 |
| Figure 7-404 | MRBL Locator Register | 1522 |
| Figure 8-1 | Tx Test Board for Non-EMBEDDED RefClk | 1524 |
| Figure 8-2 | Tx Test board for Embedded RefClk..... | 1524 |
| Figure 8-3 | Single-ended and Differential Levels | 1526 |
| Figure 8-4 | Tx Equalization FIR Representation for 8.0, 16.0, and 32.0 GT/s | 1527 |
| Figure 8-5 | Tx Equalization FIR Representation for 64.0 GT/s | 1528 |
| Figure 8-6 | Definition of Tx Voltage Levels and Equalization Ratios..... | 1529 |
| Figure 8-7 | Methodology for measuring Tx equalization coefficients and presets | 1531 |
| Figure 8-8 | V _{TX-DIFF-PP} and V _{TX-DIFF-PP-LOW} Measurement | 1532 |
| Figure 8-9 | Transmit Equalization Coefficient Space Triangular Matrix Example for 8.0, 16.0, and 32.0 GT/s..... | 1533 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Figure 8-10 | Transmit Equalization Coefficient Space Triangular Matrix Example for 64.0 GT/s | 1534 |
| Figure 8-11 | Measuring $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ at 8.0 GT/s | 1536 |
| Figure 8-12 | Compliance Pattern and Resulting Package Loss Test Waveform | 1537 |
| Figure 8-13 | Example of Normalized Four Symbol Linear Pulse Responses..... | 1545 |
| Figure 8-14 | Example of Un-normalized Four Symbol Linear Pulse Responses | 1546 |
| Figure 8-15 | 2.5 and 5.0 GT/s Transmitter Margining Voltage Levels and Codes..... | 1547 |
| Figure 8-16 | First Order CC Behavioral CDR Transfer Functions | 1551 |
| Figure 8-17 | 2 nd Order Behavioral SRIS CDR Transfer Functions for 2.5 GT/s and 5.0 GT/s | 1552 |
| Figure 8-18 | Behavioral SRIS CDR Function for 8.0 GT/s, and SRIS and CC CDR for 16.0 and 32.0 GT/s..... | 1553 |
| Figure 8-19 | Behavioral SRIS and CC CDR for 64.0 GT/s | 1554 |
| Figure 8-20 | Relation Between Data Edge PDFs and Recovered Data Clock | 1556 |
| Figure 8-21 | Derivation of T_{TX-UTJ} and $T_{TX-UDJDD}$ | 1556 |
| Figure 8-22 | PWJ Relative to Consecutive Edges 1 UI Apart..... | 1557 |
| Figure 8-23 | Definition of $T_{TX-UPW-DJDD}$ and $T_{TX-UPW-TJ}$ Data Rate Dependent Transmitter Parameters..... | 1557 |
| Figure 8-24 | Tx, Rx Differential Return Loss Mask with 50 Ohm Reference | 1562 |
| Figure 8-25 | Tx, Rx Common Mode Return Loss Mask with 50 Ohm Reference | 1563 |
| Figure 8-26 | 64.0 GT/s Tx, Rx Differential Return Loss Mask with 50 Ohm Reference | 1564 |
| Figure 8-27 | 64.0 GT/s Tx, Rx Common Mode Return Loss Mask with 50 Ohm Reference | 1565 |
| Figure 8-28 | Rx Test board Topology for 16.0 and 32.0 GT/s | 1568 |
| Figure 8-29 | Example Calibration Channel IL Mask Excluding Rx Package for 8.0 GT/s | 1569 |
| Figure 8-30 | Example 16.0 GT/s Calibration Channel | 1573 |
| Figure 8-31 | Stackup for Example 16.0 GT/s Calibration Channel | 1573 |
| Figure 8-32 | CEM Connector Drill Hole Pad Stack | 1574 |
| Figure 8-33 | Pad Stack for SMA Drill Holes..... | 1575 |
| Figure 8-34 | Example 32.0 GT/s Calibration Channel | 1577 |
| Figure 8-35 | Stack-up for Example 32.0 GT/s Calibration Channel | 1577 |
| Figure 8-36 | Transfer Function for 8.0 GT/s Behavioral CTLE..... | 1579 |
| Figure 8-37 | Loss Curves for 8.0 GT/s Behavioral CTLE | 1580 |
| Figure 8-38 | Loss Curves for 16.0 GT/s Behavioral CTLE | 1580 |
| Figure 8-39 | Loss Curves for 32.0 GT/s Behavioral CTLE | 1582 |
| Figure 8-40 | Loss Curves for 64.0 GT/s Behavioral CTLE | 1584 |
| Figure 8-41 | Variables Definition and Diagram for 1-tap DFE | 1585 |
| Figure 8-42 | Diagram for 2-tap DFE | 1585 |
| Figure 8-43 | Layout for Calibrating the Stressed Jitter Eye at 8.0 GT/s | 1589 |
| Figure 8-44 | Layout for Calibrating the Stressed Jitter Eye at 16.0, 32.0, and 64.0 GT/s..... | 1590 |
| Figure 8-45 | Sj Mask for Receivers Operating in IR mode at 8.0 GT/s..... | 1593 |
| Figure 8-46 | Sj Mask for Receivers Operating in SRIS mode at 16.0 GT/s | 1594 |
| Figure 8-47 | Sj Mask for Receivers Operating in CC mode at 16.0 GT/s | 1595 |
| Figure 8-48 | Sj Mask for Receivers Operating in SRIS mode at 32.0 GT/s | 1596 |
| Figure 8-49 | Sj Mask for Receivers Operating in CC mode at 32.0 GT/s | 1597 |
| Figure 8-50 | Sj Mask for Receivers Operating in SRIS mode at 64.0 GT/s | 1598 |
| Figure 8-51 | Sj Mask for Receivers Operating in CC mode at 64.0 GT/s | 1599 |
| Figure 8-52 | Sj Masks for Receivers Operating in CC Mode at 8.0 GT/s | 1600 |
| Figure 8-53 | Layout for Jitter Testing Common Refclk Rx at 16.0 GT/s..... | 1601 |
| Figure 8-54 | Layout for Jitter Testing for Independent Refclk Rx at 16.0 GT/s | 1601 |
| Figure 8-55 | Exit from Idle Voltage and Time Margins..... | 1604 |
| Figure 8-56 | Allowed Ranges for Maximum NRZ Timing and Voltage Margin | 1605 |
| Figure 8-57 | Allowed Ranges for Maximum PAM4 Timing and Voltage Margins | 1606 |
| Figure 8-58 | Flow Diagram for Channel Tolerancing at 2.5 and 5.0 GT/s..... | 1611 |
| Figure 8-59 | Flow Diagram for Channel Tolerancing at 8.0 and 16.0 GT/s..... | 1611 |
| Figure 8-60 | Tx/Rx Behavioral Package Models | 1612 |
| Figure 8-61 | Behavioral Tx and Rx S-Port Designation for 8.0 and 16.0 GT/s Packages | 1613 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Figure 8-62 | SDD21 Plots for Root and Non-Root Packages for 16.0 GT/s | 1613 |
| Figure 8-63 | Insertion Loss for Root Reference Package for 32.0 GT/s..... | 1614 |
| Figure 8-64 | Return Loss for Root Reference Package for 32.0 GT/s | 1614 |
| Figure 8-65 | NEXT for Root Reference Package (Worst-Case) for 32.0 GT/s..... | 1615 |
| Figure 8-66 | FEXT for Root Reference Package (Worst-Case) for 32.0 GT/s..... | 1615 |
| Figure 8-67 | Insertion Loss for Non-Root Reference Package for 32.0 GT/s | 1616 |
| Figure 8-68 | Return Loss for Non-Root Reference Package for 32.0 GT/s | 1616 |
| Figure 8-69 | NEXT for Non-Root Reference Package (Worst-Case) for 32.0 GT/s..... | 1617 |
| Figure 8-70 | FEXT for Non-Root Reference Package (Worst-Case) for 32.0 GT/s | 1617 |
| Figure 8-71 | Insertion Loss for Root Reference Package for 64.0 GT/s..... | 1618 |
| Figure 8-72 | Return Loss for Root Reference Package for 64.0 GT/s | 1618 |
| Figure 8-73 | NEXT for Root Reference Package (Worst Case) for 64.0 GT/s | 1619 |
| Figure 8-74 | FEXT for Root Reference Package (Worst Case) for 64.0 GT/s..... | 1619 |
| Figure 8-75 | Insertion Loss for Non-Root Reference Package for 64.0 GT/s | 1620 |
| Figure 8-76 | Return Loss for Non-Root Reference Package for 64.0 GT/s | 1620 |
| Figure 8-77 | NEXT for Non-Root Reference Package (Worst Case) for 64.0 GT/s | 1621 |
| Figure 8-78 | FEXT for Non-Root Reference Package (Worst Case) for 64.0 GT/s..... | 1621 |
| Figure 8-79 | 32.0 and 64.0 GT/s Reference Package Port Connections for Pin to Pin Channel Evaluation | 1622 |
| Figure 8-80 | Example Derivation of 8.0 GT/s Jitter Parameters for | 1624 |
| Figure 8-81 | EH, EW Mask | 1627 |
| Figure 8-82 | Oscilloscope Refclk Test Setup for All Cases Except Jitter at 32.0 and 64.0 GT/s | 1631 |
| Figure 8-83 | Single-Ended Measurement Points for Absolute Cross Point and Swing | 1633 |
| Figure 8-84 | Single-Ended Measurement Points for Delta Cross Point | 1633 |
| Figure 8-85 | Single-Ended Measurement Points for Rise and Fall Time Matching | 1633 |
| Figure 8-86 | Differential Measurement Points for Duty Cycle and Period | 1634 |
| Figure 8-87 | Differential Measurement Points for Rise and Fall Time | 1634 |
| Figure 8-88 | Differential Measurement Points for Ringback | 1634 |
| Figure 8-89 | Limits for phase jitter from the Reference with 5000 ppm SSC..... | 1636 |
| Figure 8-90 | 5 MHz PLL Transfer Function Example | 1637 |
| Figure 8-91 | Common Refclk Rx Architecture for all Data Rates Except 32.0 and 64.0 GT/s | 1638 |
| Figure 8-92 | Common Refclk PLL and CDR Characteristics for 2.5 GT/s | 1639 |
| Figure 8-93 | Common Refclk PLL and CDR Characteristics for 5.0 GT/s | 1640 |
| Figure 8-94 | Common Refclk PLL and CDR Characteristics for 8.0 and 16.0 GT/s..... | 1640 |
| Figure 8-95 | Common Refclk PLL and CDR Characteristics for 32.0 GT/s | 1640 |
| Figure 8-96 | Common Refclk PLL and CDR Characteristics for 64.0 GT/s | 1640 |
| Figure 9-1 | Generic Platform Configuration | 1644 |
| Figure 9-2 | Generic Platform Configuration with a VI and Multiple SI | 1645 |
| Figure 9-3 | Generic Platform Configuration with SR-IOV and IOV Enablers | 1646 |
| Figure 9-4 | Example Multi-Function Device | 1648 |
| Figure 9-5 | Example SR-IOV Single PF Capable Device | 1649 |
| Figure 9-6 | Example SR-IOV Multi-PF Capable Device | 1651 |
| Figure 9-7 | Example ↑↓SR-IOV Device↑ ↓↑SR-IOV Device↓ with Multiple Bus Numbers | 1653 |
| Figure 9-8 | Example ↑↓SR-IOV Device↑ ↓↑SR-IOV Device↓ with a Mixture of Function Types..... | 1654 |
| Figure 9-9 | ↑↓Example SIOV Capable Device↓ | 1655 |
| Figure 9-10 | BAR Space Example for Single BAR Device | 1660 |
| Figure 9-11 | SR-IOV Extended Capability | 1667 |
| Figure 9-12 | SR-IOV Extended Capability Header | 1668 |
| Figure 9-13 | SR-IOV Capabilities Register | 1668 |
| Figure 9-14 | SR-IOV Control Register..... | 1671 |
| Figure 9-15 | SR-IOV Status | 1675 |
| Figure 9-16 | ↑↓SIOV Extended Capability↓ | 1685 |

 ECN: Base 6.3
 SIOV△↔

Base 6.4 vs Base 6.3

| | | |
|--------------|---|------|
| Figure 9-17 | EditorialSIOV Extended Capability Header | 1685 |
| Figure 9-18 | EditorialSIOV Capabilities Register..... | 1686 |
| Figure 9-19 | EditorialSIOV Status Register | 1687 |
| Figure 10-1 | Example Illustrating a Platform with TA, ATPT , and ATC Elements | 1690 |
| Figure 10-2 | Example ATS Translation Request/Completion Exchange | 1690 |
| Figure 10-3 | Example Multi-Function Device with ATC per Function..... | 1693 |
| Figure 10-4 | Invalidation Protocol with a Single Invalidate Request and Completion..... | 1693 |
| Figure 10-5 | Single Invalidate Request with Multiple Invalidate Completions | 1694 |
| Figure 10-6 | Memory Request Header with 64-bit Address Highlighting AT field..... | 1697 |
| Figure 10-7 | Memory Request Header with 32-bit Address Highlighting AT field | 1697 |
| Figure 10-8 | Memory Request Header with 64-bit Address Highlighting AT field - Flit Mode | 1697 |
| Figure 10-9 | Memory Request Header with 32-bit Address Highlighting AT field - Flit Mode | 1698 |
| Figure 10-10 | Translation Request with 64-bit Address - Non-Flit Mode..... | 1699 |
| Figure 10-11 | Translation Request with 32-bit Address - Non-Flit Mode..... | 1699 |
| Figure 10-12 | Translation Request with 64-bit Address - Flit Mode | 1699 |
| Figure 10-13 | Translation Request with 32-bit Address - Flit Mode | 1700 |
| Figure 10-14 | Translation Completion Data Entry..... | 1703 |
| Figure 10-15 | Example Translation Completion with 1 TLP..... | 1711 |
| Figure 10-16 | Example Translation Completion with 2 TLPs | 1712 |
| Figure 10-17 | ATS Memory Attributes Example | 1714 |
| Figure 10-18 | Invalidate Request Message - Non-Flit Mode | 1716 |
| Figure 10-19 | Invalidate Request Message - Flit Mode | 1717 |
| Figure 10-20 | Invalidate Request Message Body | 1718 |
| Figure 10-21 | Invalidate Completion Message Format - Non-Flit Mode..... | 1719 |
| Figure 10-22 | Invalidate Completion Message - Flit Mode | 1719 |
| Figure 10-23 | Page Request Message - Non-Flit Mode | 1726 |
| Figure 10-24 | Page Request Message - Flit Mode..... | 1726 |
| Figure 10-25 | Stop Marker Message - Non-Flit Mode..... | 1729 |
| Figure 10-26 | Stop Marker Message - Flit Mode | 1729 |
| Figure 10-27 | PRG Response Message - Non-Flit Mode | 1731 |
| Figure 10-28 | PRG Response Message - Flit Mode | 1731 |
| Figure 10-29 | ATS Extended Capability Structure | 1733 |
| Figure 10-30 | ATS Extended Capability Header | 1733 |
| Figure 10-31 | ATS Capability Register (Offset 04h) | 1734 |
| Figure 10-32 | ATS Control Register..... | 1735 |
| Figure 10-33 | Page Request Extended Capability Structure | 1736 |
| Figure 10-34 | Page Request Extended Capability Header..... | 1736 |
| Figure 10-35 | Page Request Control Register | 1737 |
| Figure 10-36 | Page Request Status Register | 1738 |
| Figure 11-1 | Conceptual View with Example Host and Device and Logical Communication Paths..... | 1742 |
| Figure 11-2 | TDISP Host/Device Reference Architecture..... | 1745 |
| Figure 11-3 | Identification of Requests | 1748 |
| Figure 11-4 | TDI Identifier – INTERFACE_ID | 1749 |
| Figure 11-5 | TDISP State Machine | 1749 |
| Figure 11-6 | TDISP Request/Response Encapsulation..... | 1756 |
| Figure 11-7 | Example Flow Where DSM is Unable to Return Full Length Report | 1771 |
| Figure 12-1 | Example PESTI Application | 1803 |
| Figure 12-2 | UART Data Framing | 1803 |
| Figure 12-3 | PESTI Circuit Diagram | 1804 |
| Figure 12-4 | PESTI Broadcast Command..... | 1808 |
| Figure 12-5 | PESTI Protocol Phases | 1810 |
| Figure 12-6 | PESTI Discovery Command and Response Format | 1812 |

Base 6.4 vs Base 6.3

| | | |
|--------------|--|------|
| Figure 12-7 | Single Byte PESTI Virtual Wire Exchange..... | 1814 |
| Figure 12-8 | Multi-byte PESTI Virtual Wire Exchange | 1814 |
| Figure 12-9 | PESTI Fan-out Methods..... | 1818 |
| Figure 12-10 | PESTI Mux Switch Control Format..... | 1819 |
| Figure 12-11 | [CEM] form factor example circuit for repurposing legacy JTAG to USB 2.0 mode..... | 1821 |
| Figure 12-12 | Example of 2-wire, 8-bit addressing for a card carrier with N end form factors in SMBus mode | 1824 |
| Figure 12-13 | Example of ↑↓2-wire↓ <ins>↑↓2-Wire↑</ins> Hub Use | 1826 |
| Figure 12-14 | SMBus to I3C Transition Flow | 1829 |
| Figure 12-15 | Component Timing Diagram for Transition to I3C Signaling Voltage..... | 1831 |
| Figure 12-16 | SMBus/I2C-based FRU Information Device Writes with Two-Byte Addressing | 1834 |
| Figure 12-17 | FRU Information Device Reads with Two-Byte Addressing | 1834 |
| Figure 12-18 | <ins>↑↓PCIe-MI Component↑</ins> Protocol ↑↓(MCTP),↓ <ins>↑↓Layers↑</ins> | 1841 |
| Figure 12-19 | <ins>↑↓Example PCIe-MI Component↑</ins> | 1848 |
| Figure 12-20 | <ins>↑↓Example PCIe-MI Message Flows↑</ins> | 1851 |
| Figure 12-21 | <ins>↑↓PCIe-MI Message Formats↑</ins> | 1852 |
| Figure 12-22 | Example Tiers Involving Sidebands..... | 1885 |
| Figure A-1 | An Example Showing Endpoint-to-Root-Complex and Peer-to-Peer Communication Models | 1887 |
| Figure A-2 | Two Basic Bandwidth Resourcing Problems: Over-Subscription and Congestion | 1888 |
| Figure A-3 | A Simplified Example Illustrating PCI Express Isochronous Parameters | 1892 |
| Figure C-1 | Scrambling Spectrum at 2.5 GT/s for Data Value of 0 | 1912 |
| Figure E-1 | Reference Topology for IDO Use | 1919 |
| Figure G-1 | Device and Processor Connected Using a PMUX Link | 1927 |
| Figure G-2 | PMUX Link..... | 1927 |
| Figure G-3 | PMUX Packet Flow Through the Layers..... | 1928 |
| Figure G-4 | PMUX Packet..... | 1932 |
| Figure G-5 | TLP and PMUX Packet Framing (8b/10b Encoding) | 1933 |
| Figure G-6 | TLP and PMUX Packet Framing (128b/130b Encoding) | 1935 |
| Figure G-7 | PMUX Extended Capability..... | 1938 |
| Figure G-8 | PMUX Extended Capability Header..... | 1938 |
| Figure G-9 | PMUX Capability Register..... | 1939 |
| Figure G-10 | PMUX Control Register | 1940 |
| Figure G-11 | PMUX Status Register | 1941 |
| Figure G-12 | PMUX Protocol Array Entry | 1942 |
| Figure L-1 | Example Memory Write TLP (NFM) | 2257 |
| Figure L-2 | Example NFM Memory Write IDE TLP | 2259 |
| Figure L-3 | Example Memory Write TLP (FM)..... | 2261 |
| Figure L-4 | Example Memory Write IDE TLP (FM) | 2263 |
| Figure L-5 | Example Memory Write TLP with Partial Header Encryption (NFM) | 2264 |
| Figure L-6 | Example NFM Memory Write IDE TLP with Partial Header Encryption | 2265 |
| Figure L-7 | Example Memory Write TLP with Partial Header Encryption (FM) | 2268 |
| Figure L-8 | Example Memory Write IDE TLP with Partial Header Encryption (FM) | 2270 |

Base 6.4 vs Base 6.3

List of Equations

| | | |
|---------------|--|------|
| Equation 2-1 | CREDITS_CONSUMED | 324 |
| Equation 2-2 | SHARED_CREDITS_CONSUMED..... | 324 |
| Equation 2-3 | SUM_SHARED_CREDITS_CONSUMED | 325 |
| Equation 2-4 | TLP SHARED_CREDITS_CONSUMED_CURRENTLY..... | 325 |
| Equation 2-5 | FC SHARED_CREDITS_CONSUMED_CURRENTLY..... | 325 |
| Equation 2-6 | SUM_SHARED_CREDIT_LIMIT..... | 327 |
| Equation 2-7 | SHARED_CUMULATIVE_CREDITS_REQUIRED | 327 |
| Equation 2-8 | Shared Transmitter Gate non-[Merged] | 327 |
| Equation 2-9 | Shared Transmitter Gate [Merged] | 328 |
| Equation 2-10 | Shared Transmitter Usage Limit Gate non-[Merged]..... | 328 |
| Equation 2-11 | Shared Transmitter Usage Limit Gate [Merged]..... | 328 |
| Equation 2-12 | CUMULATIVE_CREDITS_REQUIRED | 329 |
| Equation 2-13 | Transmitter Gate..... | 329 |
| Equation 2-14 | CREDITS_ALLOCATED..... | 330 |
| Equation 2-15 | CREDITS_RECEIVED..... | 330 |
| Equation 2-16 | Receiver Overflow Error Check Non-Flit / Dedicated..... | 331 |
| Equation 2-17 | Receiver Overflow Error Check Non-Posted / Not [Merged]..... | 331 |
| Equation 2-18 | Receiver Overflow Error Check [Merged] | 332 |
| Equation 3-1 | Tx SEQ Stall..... | 378 |
| Equation 3-2 | Tx SEQ Update..... | 379 |
| Equation 4-1 | Parity bytes | 495 |
| Equation 4-2 | Check bytes | 495 |
| Equation 4-3 | Retimer Latency with SRIS..... | 728 |
| Equation 6-1 | MC_Overlay Transform rules | 910 |
| Equation 6-2 | PTM Master Time | 936 |
| Equation 7-1 | MSI-X Starting Address | 1200 |
| Equation 7-2 | MSI-X PBA QWORD Access..... | 1200 |
| Equation 7-3 | MSI-X PBA DWORD Access | 1200 |
| Equation 7-4 | Egress Control Vector Access | 1269 |
| Equation 8-1 | $V_{\text{DIFFp-p}}$ | 1525 |
| Equation 8-2 | $V_{\text{TX-AC-CM-PP}}$ | 1525 |
| Equation 8-3 | Y | 1538 |
| Equation 8-4 | X_r | 1539 |
| Equation 8-5 | X | 1539 |
| Equation 8-6 | P | 1539 |
| Equation 8-7 | E | 1539 |
| Equation 8-8 | y | 1540 |
| Equation 8-9 | X_{NP} | 1540 |
| Equation 8-10 | P_{EO} | 1540 |
| Equation 8-11 | e | 1541 |
| Equation 8-12 | $\sigma_{L,i}$ | 1541 |
| Equation 8-13 | $\mu_{L,i}$ | 1541 |
| Equation 8-14 | σ_L | 1542 |
| Equation 8-15 | σ_n | 1542 |
| Equation 8-16 | SNDR | 1542 |
| Equation 8-17 | R_{LM} | 1543 |

| | | |
|---------------|---|------|
| Equation 8-18 | W_{sym0_3} | 1544 |
| Equation 8-19 | X_{sym} | 1544 |
| Equation 8-20 | P_{sym} | 1544 |
| Equation 8-21 | $P_{sym_un_normalized}$ | 1546 |
| Equation 8-22 | Behavioral SRIS CDR at 8.0 GT/s and SRIS and CC Behavioral CDR at 16.0 GT/s | 1553 |
| Equation 8-23 | SRIS Behavioral CDR Parameters at 8.0 GT/s | 1553 |
| Equation 8-24 | SRIS and CC Behavioral CDR Parameters at 16.0 GT/s..... | 1554 |
| Equation 8-25 | SRIS and CC Behavioral CDR Parameters at 32.0 and 64.0 GT/s..... | 1555 |
| Equation 8-26 | Behavioral CTLE at 32.0 GT/s | 1581 |
| Equation 8-27 | Behavioral CTLE at 64.0 GT/s | 1583 |
| Equation 8-28 | Relationship between 2 nd order PLL natural frequency and 3 dB point | 1637 |
| Equation A-1 | Isochronous Bandwidth..... | 1888 |
| Equation A-2 | Isochronous Payload Size | 1889 |
| Equation A-3 | N_{max} | 1890 |
| Equation A-4 | BW_{max} | 1890 |
| Equation A-5 | $BW_{granularity}$ | 1890 |
| Equation A-6 | N_{link} | 1890 |
| Equation A-7 | Max Isochronous Transaction Latency | 1891 |
| Equation H-1 | Max UpdateFC Latency | 1945 |
| Equation H-2 | Max Ack Latency | 1948 |

List of Tables

| | | |
|------------|--|-----|
| Table 1-1 | PCIe Signaling Characteristics | 143 |
| Table 2-1 | Transaction Types for Different Address Spaces..... | 156 |
| Table 2-2 | Fmt[2:0] Field Values..... | 160 |
| Table 2-3 | ↑↓Non-Flit Mode↑ Fmt[2:0] and Type[4:0] Field Encodings..... | 161 |
| Table 2-4 | Length[9:0] Field Encoding | 162 |
| Table 2-5 | Flit Mode TLP Header Type Encodings..... | 164 |
| Table 2-6 | ↑↓OHC A↓ ↑↓OHC-Ax↑ Included Fields for OHC-A1 through OHC-A5 (see through) | 175 |
| Table 2-7 | Address Field Mapping..... | 196 |
| Table 2-8 | Header Field Locations for non-ARI ID Routing - Non-Flit Mode..... | 198 |
| Table 2-9 | Header Field Locations for ARI ID Routing ↑↓- Non-Flit Mode↑ | 198 |
| Table 2-10 | Byte Enables Location and Correspondence | 209 |
| Table 2-11 | Tag Enables, Sizes, and Permitted Ranges for non-UIO Transactions | 215 |
| Table 2-12 | Ordering Attributes | 220 |
| Table 2-13 | Cache Coherency Management Attribute | 221 |
| Table 2-14 | Definition of TC Field Encodings..... | 221 |
| Table 2-15 | Length Field Values for AtomicOp Requests | 222 |
| Table 2-16 | TPH TLP Prefix Bit Mapping ↑↓- Non-Flit Mode↑ | 228 |
| Table 2-17 | Location of PH[1:0] in TLP Header ↑↓- Non-Flit Mode↑ | 230 |
| Table 2-18 | Processing Hint Encoding | 230 |
| Table 2-19 | Location of ST[7:0] in TLP Headers ↑↓- Non-Flit Mode↑ | 232 |
| Table 2-20 | Message Routing | 237 |
| Table 2-21 | INTx Mechanism Messages | 238 |
| Table 2-22 | Bridge Mapping for INTx Virtual Wires..... | 240 |
| Table 2-23 | Power Management Messages | 242 |
| Table 2-24 | Error Signaling Messages | 243 |
| Table 2-25 | ERR_COR Subclass (ECS) Field Encodings..... | 244 |
| Table 2-26 | Unlock Message..... | 245 |
| Table 2-27 | Set_Slot_Power_Limit Message | 245 |
| Table 2-28 | Vendor-Defined Messages..... | 246 |
| Table 2-29 | DRS Message..... | 250 |
| Table 2-30 | FRS Message | 251 |
| Table 2-31 | Hierarchy ID Message | 253 |
| Table 2-32 | Ignored Messages..... | 254 |
| Table 2-33 | LTR Message..... | 255 |
| Table 2-34 | OBFF Message | 257 |
| Table 2-35 | Precision Time Measurement Messages | 259 |
| Table 2-36 | IDE Messages | 262 |
| Table 2-37 | Completion Status Field Values..... | 269 |
| Table 2-38 | Local TLP Prefix Types | 275 |
| Table 2-39 | End-End TLP Prefix Types | 277 |
| Table 2-40 | Calculating Byte Count from Length and ↑↓First/Last↑ Byte Enables | 293 |
| Table 2-41 | Calculating Lower Address from First DW ↑↓BE↓ ↑↓Byte Enables↑ | 294 |
| Table 2-42 | ↑↓Non-UIO↑ Ordering Rules ↑↓Summary↓ | 299 |
| Table 2-43 | UIO TLP Ordering Rules | 304 |
| Table 2-44 | UIO Acceptance Dependency Rules – Downstream Ports | 304 |
| Table 2-45 | UIO Acceptance Dependency Rules – Upstream Ports | 305 |
| Table 2-46 | Streamlined ↑↓VC↓ ↑↓Virtual Channel↑ (SVC) TC/VC Default Assignments | 308 |

Base 6.4 vs Base 6.3

| | | |
|------------|---|----------------------------|
| Table 2-47 | TC to VC Mapping Example | 312 |
| Table 2-48 | Flow Control Credit Types | 316 |
| Table 2-49 | TLP Flow Control Credit Consumption | 316 |
| Table 2-50 | Minimum Initial Flow Control Advertisements | 318 |
| Table 2-51 | [Field Size] Values..... | 321 |
| Table 2-52 | Maximum UpdateFC Transmission Latency Guidelines for 2.5 GT/s (Symbol Times)..... | 335 |
| Table 2-53 | Maximum UpdateFC Transmission Latency Guidelines for 5.0 GT/s (Symbol Times)..... | 335 |
| Table 2-54 | Maximum UpdateFC Transmission Latency Guidelines for 8.0 GT/s and Higher Data Rates (Symbol Times) | 336 |
| Table 2-55 | Mapping of Bits into ECRC Field | 337 |
| Table 3-1 | Data Link Feature Supported Bit Definition..... | 355 |
| Table 3-2 | InitFC1 / InitFC2 Options – Non-Flit Mode..... | 358 |
| Table 3-3 | InitFC1 / InitFC2 Options – Flit Mode..... | 358 |
| Table 3-4 | Scaled Flow Control Scaling Factors | 364 |
| Table 3-5 | DLLP Type Encodings..... | 366 |
| Table 3-6 | HdrScale and DataScale Encodings | 369 |
| Table 3-7 | Mapping of Bits into CRC Field | 375 |
| Table 3-8 | Mapping of Bits into LCRC Field..... | 379 |
| Table 3-10 | Maximum Ack Latency Limits for 2.5 GT/s (Symbol Times) (-0%/+0%) | 390 |
| Table 3-11 | Maximum Ack Latency Limits for 5.0 GT/s (Symbol Times) (-0%/+0%) | 390 |
| Table 3-12 | Maximum Ack Latency Limits for 8.0 GT/s and higher data rates (Symbol Times)..... | 390 |
| Table 4-1 | Valid Encoding and Data Stream Mode Combinations..... | 394 |
| Table 4-2 | Valid Encoding for Ordered Sets..... | 394 |
| Table 4-3 | Special Symbols in 8b/10b Encoding | 397 |
| Table 4-4 | Framing Token Encoding | 407 |
| Table 4-6 | Effect of +/-1 voltage level error on the wire for various PAM4 voltage levels – at most one bit flips with an error on a UI..... | 429 |
| Table 4-7 | Truth Table for Precoding on the Transmit side | 429 |
| Table 4-8 | Truth Table for Precoding on the Receive side | 430 |
| Table 4-9 | Example of precoding with an error in the channel and DFE error propagation at the Receiver..... | 431 |
| Table 4-10 | Flit Layout in a x16 Link | 437 |
| Table 4-11 | Flit interleaving in a x8 Link | 438 |
| Table 4-12 | Flit interleaving in a x4 Link | 439 |
| Table 4-13 | Flit interleaving in a x2 Link | 439 |
| Table 4-14 | Flit arrangement in a x1 Link | 440 |
| Table 4-15 | Example TLP Placement in Flit Mode on a x16 Link | 444 |
| Table 4-16 | Flit Types..... | 446 |
| Table 4-17 | DLP Bytes in the Flit | 446 |
| Table 4-18 | Optimized_Update_FC..... | 447 |
| Table 4-19 | Flit_Marker..... | 448 |
| Table 4-20 | ↑↑Nak Ignore Window↑↑ | 454 |
| | | Errata: Base 6.3 B837△▷ |
| Table 4-21 | ↑↑Ack/Nak Latency↑↑ | 454 |
| | | Errata: Base 6.3 B837△◁ |
| Table 4-22 | ↑↑NOP Flit Types↑↑ | 474 |
| Table 4-23 | ↑↑NOP Flit Common Header Fields↑↑ | 475 |
| Table 4-24 | NOP.Debug Flit Debug Chunk Fields | 478 |
| Table 4-25 | PCI-SIG Defined Debug Chunk Opcode Values | 481 |
| Table 4-26 | FC Information Tracked by Transmitter Encodings | 485 |
| Table 4-27 | FC Information Tracked by Receiver Encodings | 487 |
| Table 4-28 | Flit Mode Transmitter Retry Flags and Counters Fields..... | 489 |

Base 6.4 vs Base 6.3

| | | |
|------------|---|-----|
| Table 4-29 | Flit Mode Receiver Retry Flags and Counters Fields | 491 |
| Table 4-30 | Buffer Occupancy Encodings | 492 |
| Table 4-32 | Ordered Set insertion interval once Data Stream starts in terms of number of Flits | 503 |
| Table 4-33 | Equalization Requirements Under Different Conditions | 508 |
| Table 4-34 | Transmitter Preset Encoding | 516 |
| Table 4-35 | Receiver Preset Hint Encoding for 8.0 GT/s | 517 |
| Table 4-36 | TS1 Ordered Set in 8b/10b and 128b/130b Encoding | 521 |
| Table 4-37 | TS2 Ordered Set in 8b/10b and 128b/130b Encoding | 528 |
| Table 4-38 | Modified TS1/TS2 Ordered Set (8b/10b encoding) | 531 |
| Table 4-39 | TS1/TS2 Ordered Set with 1b/1b Encoding | 534 |
| Table 4-40 | TS0 Ordered Set | 536 |
| Table 4-41 | Modified TS Information 1 field in Modified TS1/TS2 Ordered Sets if Modified TS Usage = 010b (Alternate Protocol) | 541 |
| Table 4-42 | Electrical Idle Ordered Set (EIOS) for 2.5 GT/s and 5.0 GT/s Data Rates | 543 |
| Table 4-43 | Electrical Idle Ordered Set (EIOS) for 128b/130b Encoding | 543 |
| Table 4-44 | Electrical Idle Ordered Set (EIOS) for 1b/1b Encoding | 543 |
| Table 4-45 | Electrical Idle Exit Ordered Set (EIEOS) for 5.0 GT/s Data Rate | 543 |
| Table 4-46 | Electrical Idle Exit Ordered Set (EIEOS) for 8.0 GT/s Data Rate | 544 |
| Table 4-47 | Electrical Idle Exit Ordered Set (EIEOS) for 16.0 GT/s Data Rate | 544 |
| Table 4-48 | Electrical Idle Exit Ordered Set (EIEOS) for 32.0 GT/s Data Rate | 544 |
| Table 4-49 | Electrical Idle Exit Ordered Set (EIEOS) for 64.0 GT/s Data Rate | 544 |
| Table 4-50 | Electrical Idle Inference Conditions | 548 |
| Table 4-51 | FTS for 8.0 GT/s and Above Data Rates | 550 |
| Table 4-52 | SDS Ordered Set (for 8.0 GT/s and 16.0 GT/s Data Rate) | 551 |
| Table 4-53 | SDS Ordered Set (for 32.0 GT/s) | 551 |
| Table 4-54 | SDS Ordered Set (for 64.0 GT/s) | 551 |
| Table 4-55 | Summary of L0p Transmitter/Receiver Behavior | 562 |
| Table 4-56 | Link Management DLLP | 562 |
| Table 4-60 | Link Status Mapped to the LTSSM | 567 |
| Table 4-61 | Compliance Pattern Settings | 574 |
| Table 4-63 | Use of TS0 or TS1 Ordered Sets in different phases | 607 |
| Table 4-64 | Standard SKP Ordered Set with 128b/130b Encoding | 651 |
| Table 4-65 | Control SKP Ordered Set with 128b/130b Encoding | 652 |
| Table 4-66 | Control SKP Ordered Set with 1b/1b Encoding | 654 |
| Table 4-67 | PHY Payload for Control SKP Ordered Set with 1b/1b Encoding | 657 |
| Table 4-71 | Illustration of Modified Compliance Pattern | 665 |
| Table 4-74 | Margin Command Related Fields in the Control SKP Ordered Set | 672 |
| Table 4-75 | Margin Commands and Corresponding Responses | 676 |
| Table 4-76 | Maximum Retimer Exit Latency | 704 |
| Table 4-77 | Inferring Electrical Idle | 706 |
| Table 4-78 | Retimer Latency Limit not SRIS (Symbol times) | 727 |
| Table 4-79 | Retimer Latency Limit SRIS (Symbol times) | 727 |
| Table 5-1 | Summary of PCI Express Link Power Management States | 739 |
| Table 5-2 | Relation Between Power Management States of Link and Components | 745 |
| Table 5-3 | Encoding of the ASPM Support Field | 768 |
| Table 5-4 | Description of the Slot Clock Configuration Bit | 768 |
| Table 5-5 | Description of the Common Clock Configuration Bit | 768 |
| Table 5-6 | Encoding of the L0s Exit Latency Field | 769 |
| Table 5-7 | Encoding of the L1 Exit Latency Field | 769 |
| Table 5-8 | Encoding of the Endpoint L0s Acceptable Latency Field | 770 |
| Table 5-9 | Encoding of the Endpoint L1 Acceptable Latency Field | 770 |
| Table 5-10 | Encoding of the ASPM Control Field | 770 |

Base 6.4 vs Base 6.3

| | | |
|------------|---|------|
| Table 5-11 | L1.2 Timing Parameters | 783 |
| Table 5-12 | Aux Power Source and Availability | 784 |
| Table 5-13 | Power Management System Messages and DLLPs | 785 |
| Table 5-14 | PCI Function State Transition Delays | 786 |
| Table 6-1 | Error Messages | 804 |
| Table 6-2 | General PCI Express Error List..... | 819 |
| Table 6-3 | Physical Layer Error List..... | 819 |
| Table 6-4 | Data Link Layer Error List..... | 819 |
| Table 6-5 | Transaction Layer Error List..... | 820 |
| Table 6-6 | Multi-Function Arbitration Error Model Example..... | 844 |
| Table 6-7 | Elements of Hot-Plug | 858 |
| Table 6-8 | Attention Indicator States | 859 |
| Table 6-9 | Power Indicator States | 860 |
| Table 6-10 | Power Budgeting Deployments..... | 883 |
| Table 6-11 | ACS P2P Request Redirect and ACS P2P Egress Control Interactions..... | 898 |
| Table 6-12 | ECRC Rules for MC_Overlay | 910 |
| Table 6-13 | Processing Hint Mapping | 919 |
| Table 6-14 | ST Modes of Operation..... | 919 |
| Table 6-15 | PASID TLP Prefix | 931 |
| Table 6-16 | Emergency Power Reduction Supported Values | 951 |
| Table 6-17 | System GUID Authority ID Encoding..... | 955 |
| Table 6-19 | Small Resource Data Type Tag Bit Definitions | 970 |
| Table 6-20 | Large Resource Data Type Tag Bit Definitions | 970 |
| Table 6-21 | Resource Data Type Flags for a Typical VPD | 970 |
| Table 6-22 | Example of Add-in Serial Card Number..... | 971 |
| Table 6-23 | VPD Large and Small Resource Data Tags | 972 |
| Table 6-24 | VPD Read-Only Fields..... | 972 |
| Table 6-25 | VPD Read/Write Fields..... | 974 |
| Table 6-26 | VPD Example..... | 974 |
| Table 6-27 | NPEM States..... | 979 |
| Table 6-28 | DOE Data Object Header 1 | 986 |
| Table 6-29 | DOE Data Object Header 2 | 986 |
| Table 6-30 | DOE Discovery Request Data Object Contents (3rd DW) | 987 |
| Table 6-31 | DOE Discovery Response Data Object Contents (3rd DW) | 987 |
| Table 6-32 | DOE Discovery Response Data Object Contents (4th DW)..... | 988 |
| Table 6-33 | PCI-SIG Defined Data Object Types (Vendor ID = 0001h) | 988 |
| Table 6-34 | DOE Async Message Data Object Contents (1 DW)..... | 990 |
| Table 6-35 | TLP Types for Selective IDE Streams | 1037 |
| Table 6-36 | IDE Revised Ordering Rules for Flow-Through non-UIO IDE Streams - Per Stream..... | 1049 |
| Table 6-37 | IDE Revised Ordering Rules for Flow-Through UIO IDE Streams - Per Stream | 1049 |
| Table 6-38 | MCAP Array Register 1 | 1056 |
| Table 6-39 | MCAP Array Register 2 | 1057 |
| Table 6-40 | MCAP Header Register 1..... | 1058 |
| Table 6-41 | MCAP Header Register 2..... | 1058 |
| Table 6-42 | MCAP Header Register 3..... | 1058 |
| Table 6-43 | MCAP Header Register 4..... | 1059 |
| Table 6-44 | PCI-SIG Defined MCAP Identifiers (MCAP Vendor ID = 0001h) | 1059 |
| Table 6-45 | MMB Capabilities Register | 1062 |
| Table 6-46 | MMB Control Register..... | 1063 |
| Table 6-47 | MMB Command Register..... | 1064 |
| Table 6-48 | MMB Status Register..... | 1065 |
| Table 6-49 | MMB PCI-SIG Defined Command Return Codes (Vendor ID = 0001h) | 1066 |

Base 6.4 vs Base 6.3

| | | |
|------------|---|------|
| Table 6-50 | MMPT Capabilities Register | 1068 |
| Table 6-51 | MMPT Control Register..... | 1069 |
| Table 6-52 | MMPT Receive Message Notification Register | 1069 |
| Table 6-53 | MDVS Register Block Header Register 1 | 1070 |
| Table 6-54 | MDVS Register Block Header Register 2 | 1071 |
| Table 6-55 | MDVS Register Block Header Register 3 | 1071 |
| Table 6-56 | PCI-SIG Defined MMB Command Opcodes (Vendor ID = 0001h) | 1072 |
| Table 6-57 | MMPT Send Message Input Payload | 1073 |
| Table 6-58 | MMPT Send Message Output Payload | 1074 |
| Table 6-59 | MMPT Receive Message Input Payload..... | 1075 |
| Table 6-60 | MMPT Receive Message Output Payload..... | 1075 |
| Table 7-1 | Enhanced Configuration Address Mapping | 1083 |
| Table 7-2 | Register and Register Bit-Field Types | 1089 |
| Table 7-3 | Special Field Types to Indicate VF Attributes..... | 1091 |
| Table 7-4 | Command Register..... | 1094 |
| Table 7-5 | Status Register..... | 1096 |
| Table 7-6 | Class Code Register | 1099 |
| Table 7-7 | Header Type Register | 1100 |
| Table 7-8 | BIST Register..... | 1101 |
| Table 7-9 | Memory Base Address Register Bits 2:1 Encoding | 1105 |
| Table 7-10 | Expansion ROM Base Address Register | 1109 |
| Table 7-11 | I/O Addressing Capability | 1114 |
| Table 7-12 | Secondary Status Register | 1115 |
| Table 7-13 | Bridge Control Register | 1118 |
| Table 7-14 | Power Management Capabilities Register | 1121 |
| Table 7-15 | Power Management Control/Status Register..... | 1123 |
| Table 7-16 | Power Management Data Register | 1125 |
| Table 7-17 | Power Consumption/Dissipation Reporting | 1125 |
| Table 7-18 | PCI Express Capability List Register..... | 1128 |
| Table 7-19 | PCI Express Capabilities Register..... | 1128 |
| Table 7-20 | Device Capabilities Register..... | 1130 |
| Table 7-21 | Device Control Register | 1134 |
| Table 7-22 | Device Status Register..... | 1141 |
| Table 7-23 | Link Capabilities Register..... | 1143 |
| Table 7-24 | Link Control Register..... | 1147 |
| Table 7-26 | Link Status Register..... | 1154 |
| Table 7-27 | Slot Capabilities Register | 1156 |
| Table 7-28 | Slot Control Register | 1159 |
| Table 7-29 | Slot Status Register | 1162 |
| Table 7-30 | Root Control Register | 1164 |
| Table 7-31 | Root Capabilities Register | 1165 |
| Table 7-32 | Root Status Register | 1166 |
| Table 7-33 | Device Capabilities 2 Register | 1167 |
| Table 7-34 | Device Control 2 Register | 1172 |
| Table 7-35 | Link Capabilities 2 Register | 1176 |
| Table 7-36 | Link Control 2 Register | 1179 |
| Table 7-37 | Link Status 2 Register | 1183 |
| Table 7-38 | Slot Capabilities 2 Register | 1186 |
| Table 7-39 | PCI Express Extended Capability Header | 1188 |
| Table 7-40 | MSI Capability Header..... | 1191 |
| Table 7-41 | Message Control Register for MSI | 1192 |
| Table 7-42 | Message Address Register for MSI | 1193 |

Base 6.4 vs Base 6.3

| | | |
|------------|---|------|
| Table 7-43 | Message Upper Address Register for MSI | 1194 |
| Table 7-44 | Message Data Register for MSI | 1194 |
| Table 7-45 | Extended Message Data Register for MSI..... | 1195 |
| Table 7-46 | Mask Bits Register for MSI | 1196 |
| Table 7-47 | Pending Bits Register for MSI..... | 1196 |
| Table 7-48 | MSI-X Capability Header | 1201 |
| Table 7-49 | Message Control Register for MSI-X..... | 1201 |
| Table 7-50 | Table Offset/Table BIR Register for MSI-X..... | 1202 |
| Table 7-51 | PBA Offset/PBA BIR Register for MSI-X | 1203 |
| Table 7-52 | Message Address Register for MSI-X Table Entries..... | 1203 |
| Table 7-53 | Message Upper Address Register for MSI-X Table Entries | 1204 |
| Table 7-54 | Message Data Register for MSI-X Table Entries | 1204 |
| Table 7-55 | Vector Control Register for MSI-X Table Entries | 1205 |
| Table 7-56 | Pending Bits Register for MSI-X PBA Entries | 1206 |
| Table 7-57 | Secondary PCI Express Extended Capability Header | 1208 |
| Table 7-58 | Link Control 3 Register | 1208 |
| Table 7-59 | Lane Error Status Register | 1210 |
| Table 7-60 | Lane Equalization Control Register Entry | 1211 |
| Table 7-63 | Data Link Feature Extended Capability Header | 1213 |
| Table 7-64 | Data Link Feature Capabilities Register | 1214 |
| Table 7-65 | Data Link Feature Status Register..... | 1215 |
| Table 7-66 | Physical Layer 16.0 GT/s Extended Capability Header..... | 1218 |
| Table 7-67 | 16.0 GT/s Capabilities Register..... | 1218 |
| Table 7-68 | 16.0 GT/s Control Register..... | 1219 |
| Table 7-69 | 16.0 GT/s Status Register..... | 1219 |
| Table 7-70 | 16.0 GT/s Local Data Parity Mismatch Status Register | 1220 |
| Table 7-71 | 16.0 GT/s First Retimer Data Parity Mismatch Status Register..... | 1221 |
| Table 7-72 | 16.0 GT/s Second Retimer Data Parity Mismatch Status Register | 1222 |
| Table 7-73 | 16.0 GT/s Lane Equalization Control Register Entry | 1222 |
| Table 7-75 | Physical Layer 32.0 GT/s Extended Capability Header..... | 1225 |
| Table 7-76 | 32.0 GT/s Capabilities Register..... | 1225 |
| Table 7-77 | 32.0 GT/s Control Register | 1226 |
| Table 7-78 | 32.0 GT/s Status Register..... | 1227 |
| Table 7-79 | Received Modified TS Data 1 Register | 1229 |
| Table 7-80 | Received Modified TS Data 2 Register | 1230 |
| Table 7-81 | Transmitted Modified TS Data 1 Register | 1231 |
| Table 7-82 | Transmitted Modified TS Data 2 Register | 1232 |
| Table 7-83 | 32.0 GT/s Lane Equalization Control Register Entry | 1233 |
| Table 7-85 | Physical Layer 64.0 GT/s Extended Capability Header..... | 1235 |
| Table 7-86 | 64.0 GT/s Capabilities Register..... | 1235 |
| Table 7-87 | 64.0 GT/s Control Register | 1236 |
| Table 7-88 | 64.0 GT/s Status Register | 1236 |
| Table 7-89 | 64.0 GT/s Lane Equalization Control Register Entry | 1238 |
| Table 7-91 | Flit Logging Extended Capability Header..... | 1240 |
| Table 7-92 | Flit Error Log Interpretation | 1240 |
| Table 7-93 | Flit Error Log 1 Register | 1241 |
| Table 7-94 | Flit Error Log 2 Register | 1243 |
| Table 7-95 | Flit Error Counter Control Register | 1244 |
| Table 7-96 | Flit Error Counter Status Register | 1245 |
| Table 7-97 | FBER Measurement Control Register | 1246 |
| Table 7-98 | FBER Measurement Status 1 Register..... | 1247 |
| Table 7-99 | FBER Measurement Status 2 Register..... | 1247 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Table 7-100 | FBER Measurement Status 3 Register..... | 1248 |
| Table 7-101 | FBER Measurement Status 4 Register..... | 1248 |
| Table 7-102 | FBER Measurement Status 5 Register..... | 1249 |
| Table 7-103 | FBER Measurement Status 6 Register..... | 1249 |
| Table 7-104 | FBER Measurement Status 7 Register..... | 1250 |
| Table 7-105 | FBER Measurement Status 8 Register..... | 1250 |
| Table 7-106 | FBER Measurement Status 9 Register..... | 1250 |
| Table 7-107 | FBER Measurement Status 10 Register..... | 1251 |
| Table 7-108 | Device 3 Extended Capability Header | 1252 |
| Table 7-109 | Device Capabilities 3 Register..... | 1252 |
| Table 7-110 | Device Control 3 Register | 1255 |
| Table 7-111 | Device Status 3 Register | 1258 |
| Table 7-112 | Lane Margining at the Receiver Extended Capability Header | 1261 |
| Table 7-113 | Margining Port Capabilities Register | 1261 |
| Table 7-114 | Margining Port Status Register | 1262 |
| Table 7-115 | Lane N: Margining Control Register Entry | 1263 |
| Table 7-116 | Lane N: Margining Lane Status Register Entry | 1264 |
| Table 7-117 | ACS Extended Capability Header | 1265 |
| Table 7-118 | ACS Capability Register | 1266 |
| Table 7-119 | ACS Control Register | 1267 |
| Table 7-120 | Egress Control Vector Register..... | 1270 |
| Table 7-121 | Power Budgeting Extended Capability Header..... | 1272 |
| Table 7-122 | Power Budgeting Control Register | 1273 |
| Table 7-123 | Power Budgeting Data Register | 1275 |
| Table 7-124 | Power Budgeting Capability Register..... | 1279 |
| Table 7-125 | Power Budgeting Sense Detect Register | 1281 |
| Table 7-126 | Power Budgeting Sense Detect Encodings | 1281 |
| Table 7-127 | LTR Extended Capability Header | 1284 |
| Table 7-128 | Max Snoop Latency Register | 1285 |
| Table 7-129 | Max No-Snoop Latency Register..... | 1285 |
| Table 7-130 | ↑↑LTR Capabilities Register↑ | 1286 |
| Table 7-131 | L1 PM Substates Extended Capability Header | 1287 |
| Table 7-132 | L1 PM Substates Capabilities Register..... | 1288 |
| Table 7-133 | L1 PM Substates Control 1 Register | 1289 |
| Table 7-134 | L1 PM Substates Control 2 Register | 1291 |
| Table 7-135 | L1 PM Substates Status Register..... | 1292 |
| Table 7-136 | Advanced Error Reporting Extended Capability Header | 1295 |
| Table 7-137 | Uncorrectable Error Status Register | 1296 |
| Table 7-138 | Uncorrectable Error Mask Register | 1299 |
| Table 7-139 | Uncorrectable Error Severity Register | 1301 |
| Table 7-140 | Correctable Error Status Register | 1303 |
| Table 7-141 | Correctable Error Mask Register | 1305 |
| Table 7-142 | Advanced Error Capabilities and Control Register | 1306 |
| Table 7-143 | Header Log Register | 1308 |
| Table 7-144 | Root Error Command Register | 1309 |
| Table 7-145 | Root Error Status Register..... | 1311 |
| Table 7-146 | Error Source Identification Register | 1313 |
| Table 7-147 | TLP Prefix Log Register..... | 1314 |
| Table 7-148 | First DW of Enhanced Allocation Capability..... | 1315 |
| Table 7-149 | Second DW of Enhanced Allocation Capability..... | 1316 |
| Table 7-150 | First DW of Each Entry for Enhanced Allocation Capability | 1317 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Table 7-152 | Enhanced Allocation Entry Field Value Definitions for both the Primary Properties and Secondary Properties Fields | 1319 |
| Table 7-153 | Resizable BAR Extended Capability Header | 1323 |
| Table 7-154 | Resizable BAR Capability Register | 1324 |
| Table 7-155 | Resizable BAR Control Register | 1326 |
| Table 7-156 | VF Resizable BAR Extended Capability Header | 1330 |
| Table 7-157 | VF Resizable BAR Control Register..... | 1331 |
| Table 7-158 | ARI Extended Capability Header..... | 1332 |
| Table 7-159 | ARI Capability Register | 1333 |
| Table 7-160 | ARI Control Register | 1334 |
| Table 7-161 | PASID Extended Capability Header | 1335 |
| Table 7-162 | PASID Capability Register | 1336 |
| Table 7-163 | PASID Control Register | 1337 |
| Table 7-164 | FRS Queueing Extended Capability Header | 1338 |
| Table 7-165 | FRS Queueing Capability Register | 1339 |
| Table 7-166 | FRS Queueing Status Register... | 1340 |
| Table 7-167 | FRS Queueing Control Register | 1340 |
| Table 7-168 | FRS Message Queue Register | 1341 |
| Table 7-169 | FPB Capability Header | 1342 |
| Table 7-170 | FPB Capabilities Register | 1343 |
| Table 7-174 | FPB RID Vector Control 1 Register | 1345 |
| Table 7-177 | FPB RID Vector Control 2 Register | 1346 |
| Table 7-178 | FPB MEM Low Vector Control Register | 1347 |
| Table 7-181 | FPB MEM High Vector Control 1 Register... | 1348 |
| Table 7-184 | FPB MEM High Vector Control 2 Register... | 1350 |
| Table 7-186 | FPB Vector Access Control Register | 1351 |
| Table 7-188 | FPB Vector Access Data Register..... | 1352 |
| Table 7-189 | Flit Performance Measurement Extended Capability Header | 1353 |
| Table 7-190 | Flit Performance Measurement Capability Register | 1354 |
| Table 7-191 | Flit Performance Measurement Control Register | 1355 |
| Table 7-192 | Flit Performance Measurement Status Register | 1357 |
| Table 7-193 | LTSSM Performance Measurement Status Register | 1358 |
| Table 7-194 | Flit Error Injection Extended Capability Header | 1360 |
| Table 7-195 | Flit Error Injection Capability Register | 1361 |
| Table 7-196 | Flit Error Injection Control 1 Register..... | 1361 |
| Table 7-197 | Flit Error Injection Control 2 Register | 1363 |
| Table 7-198 | Flit Error Injection Status Register..... | 1364 |
| Table 7-199 | Ordered Set Error Injection Control 1 Register | 1365 |
| Table 7-200 | Ordered Set Error Injection Control 2 Register | 1367 |
| Table 7-201 | Ordered Set Tx Error Injection Status Register | 1367 |
| Table 7-202 | Ordered Set Rx Error Injection Status Register | 1368 |
| Table 7-203 | NOP Flit Extended Capability Header..... | 1370 |
| Table 7-204 | NOP Flit ↓↓Capabilties↓ <ins>↑↑Capabilities↑</ins> Register..... | 1371 |
| Table 7-205 | NOP Flit Control 1 Register | 1371 |
| Table 7-206 | NOP Flit Control 2 Register | 1373 |
| Table 7-207 | NOP Flit Status Register | 1374 |
| Table 7-208 | Virtual Channel Extended Capability Header..... | 1376 |
| Table 7-209 | Port VC Capability Register 1 | 1377 |
| Table 7-210 | Port VC Capability Register 2 | 1378 |
| Table 7-211 | Port VC Control Register..... | 1379 |
| Table 7-212 | Port VC Status Register..... | 1380 |
| Table 7-213 | VC Resource Capability Register | 1380 |

| | | |
|-------------|--|------|
| Table 7-214 | VC Resource Control Register | 1382 |
| Table 7-215 | VC Resource Status Register | 1384 |
| Table 7-216 | Definition of the 4-bit Entries in the VC Arbitration Table | 1385 |
| Table 7-217 | Length of the VC Arbitration Table | 1385 |
| Table 7-218 | Length of Port Arbitration Table | 1386 |
| Table 7-219 | MFVC Extended Capability Header | 1388 |
| Table 7-220 | MFVC Port VC Capability Register 1 | 1389 |
| Table 7-221 | MFVC Port VC Capability Register 2 | 1390 |
| Table 7-222 | MFVC Port VC Control Register..... | 1390 |
| Table 7-223 | MFVC Port VC Status Register..... | 1391 |
| Table 7-224 | MFVC VC Resource Capability Register..... | 1392 |
| Table 7-225 | MFVC VC Resource Control Register | 1393 |
| Table 7-226 | MFVC VC Resource Status Register | 1395 |
| Table 7-227 | Length of Function Arbitration Table | 1397 |
| Table 7-228 | Device Serial Number Extended Capability Header | 1398 |
| Table 7-229 | Serial Number Register | 1399 |
| Table 7-230 | Vendor-Specific Capability | 1400 |
| Table 7-231 | Vendor-Specific Extended Capability Header | 1401 |
| Table 7-232 | Vendor-Specific Header | 1402 |
| Table 7-233 | Designated Vendor-Specific Extended Capability Header | 1403 |
| Table 7-234 | Designated Vendor-Specific Header 1..... | 1404 |
| Table 7-235 | Designated Vendor-Specific Header 2..... | 1405 |
| Table 7-236 | RCRB Header Extended Capability Header | 1406 |
| Table 7-237 | RCRB Vendor ID and Device ID register | 1406 |
| Table 7-238 | RCRB Capabilities register | 1407 |
| Table 7-239 | RCRB Control register | 1407 |
| Table 7-240 | Root Complex Link Declaration Extended Capability Header..... | 1410 |
| Table 7-241 | Element Self Description Register | 1410 |
| Table 7-242 | Link Description Register | 1411 |
| Table 7-243 | Link Address for Link Type 1 | 1413 |
| Table 7-244 | Root Complex Internal Link Control Extended Capability Header..... | 1414 |
| Table 7-245 | Root Complex Link Capabilities Register | 1415 |
| Table 7-246 | Root Complex Link Control Register | 1418 |
| Table 7-247 | Root Complex Link Status Register | 1419 |
| Table 7-248 | Root Complex Event Collector Endpoint Association Extended Capability Header | 1420 |
| Table 7-249 | RCEC Associated Bus Numbers Register | 1421 |
| Table 7-250 | Multicast Extended Capability Header | 1423 |
| Table 7-251 | Multicast Capability Register | 1424 |
| Table 7-252 | Multicast Control Register..... | 1425 |
| Table 7-253 | MC_Base_Address Register..... | 1425 |
| Table 7-254 | MC_Receive Register | 1426 |
| Table 7-255 | MC_Block_All Register | 1427 |
| Table 7-256 | MC_Block_Untranslated Register..... | 1427 |
| Table 7-257 | MC_Overlay_BAR Register | 1428 |
| Table 7-258 | DPA Extended Capability Header..... | 1429 |
| Table 7-259 | DPA Capability Register..... | 1429 |
| Table 7-260 | DPA Latency Indicator Register..... | 1430 |
| Table 7-261 | DPA Status Register | 1431 |
| Table 7-262 | DPA Control Register | 1431 |
| Table 7-263 | Substate Power Allocation Register (0 to Substate_Max)..... | 1432 |
| Table 7-264 | TPH Requester Extended Capability Header..... | 1433 |
| Table 7-265 | TPH Requester Capability Register..... | 1434 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Table 7-266 | TPH Requester Control Register | 1435 |
| Table 7-267 | TPH ST Table Entry..... | 1436 |
| Table 7-268 | DPC Extended Capability Header | 1439 |
| Table 7-269 | DPC Capability Register | 1440 |
| Table 7-270 | DPC Control Register..... | 1441 |
| Table 7-271 | DPC Status Register..... | 1443 |
| Table 7-272 | DPC Error Source ID Register | 1444 |
| Table 7-273 | RP PIO Status Register..... | 1445 |
| Table 7-274 | RP PIO Mask Register | 1446 |
| Table 7-275 | RP PIO Severity Register | 1447 |
| Table 7-276 | RP PIO SysError Register | 1448 |
| Table 7-277 | RP PIO Exception Register..... | 1449 |
| Table 7-278 | RP PIO Header Log Register..... | 1450 |
| Table 7-279 | RP PIO ImpSpec Log Register | 1450 |
| Table 7-280 | RP PIO TLP Prefix Log Register | 1451 |
| Table 7-281 | PTM Extended Capability Header | 1452 |
| Table 7-282 | PTM Capability Register | 1453 |
| Table 7-283 | PTM Control Register..... | 1454 |
| Table 7-285 | Readiness Time Reporting Extended Capability Header..... | 1457 |
| Table 7-286 | Readiness Time Reporting 1 Register..... | 1457 |
| Table 7-287 | Readiness Time Reporting 2 Register..... | 1458 |
| Table 7-288 | Hierarchy ID Extended Capability Header | 1460 |
| Table 7-289 | Hierarchy ID Status Register | 1461 |
| Table 7-290 | Hierarchy ID Data Register | 1462 |
| Table 7-291 | Hierarchy ID GUID 1 Register..... | 1463 |
| Table 7-292 | Hierarchy ID GUID 2 Register..... | 1463 |
| Table 7-293 | Hierarchy ID GUID 3 Register..... | 1464 |
| Table 7-294 | Hierarchy ID GUID 4 Register..... | 1464 |
| Table 7-295 | Hierarchy ID GUID 5 Register..... | 1465 |
| Table 7-296 | VPD Address Register | 1467 |
| Table 7-297 | VPD Data Register..... | 1467 |
| Table 7-298 | NPEM Extended Capability Header | 1468 |
| Table 7-299 | NPEM Capability Register..... | 1469 |
| Table 7-300 | NPEM Control Register | 1470 |
| Table 7-301 | NPEM Status Register | 1472 |
| Table 7-302 | Alternate Protocol Extended Capability Header | 1473 |
| Table 7-303 | Alternate Protocol Capabilities Register | 1474 |
| Table 7-304 | Alternate Protocol Control Register..... | 1474 |
| Table 7-305 | Alternate Protocol Data 1 Register..... | 1475 |
| Table 7-306 | Alternate Protocol Data 2 Register..... | 1476 |
| Table 7-307 | Alternate Protocol Selective Enable Mask Register | 1476 |
| Table 7-308 | Advanced Features Capability Header | 1477 |
| Table 7-309 | AF Capabilities Register | 1478 |
| Table 7-310 | Conventional PCI Advanced Features Control Register..... | 1478 |
| Table 7-311 | AF Status Register..... | 1479 |
| Table 7-312 | SFI Extended Capability Header | 1480 |
| Table 7-313 | SFI Capability Register | 1481 |
| Table 7-314 | SFI Control Register..... | 1482 |
| Table 7-315 | SFI Status Register..... | 1485 |
| Table 7-316 | SFI CAM Address Register..... | 1486 |
| Table 7-317 | SFI CAM Data Register | 1486 |
| Table 7-318 | Subsystem ID and Subsystem Vendor ID Capability Header..... | 1488 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Table 7-319 | Subsystem ID and Subsystem Vendor ID Capability Data | 1488 |
| Table 7-320 | DOE Extended Capability Header | 1489 |
| Table 7-321 | DOE Capabilities Register | 1490 |
| Table 7-322 | DOE Control Register..... | 1491 |
| Table 7-323 | DOE Status Register..... | 1492 |
| Table 7-324 | DOE Write Data Mailbox Register | 1493 |
| Table 7-325 | DOE Read Data Mailbox Register | 1494 |
| Table 7-326 | ↑↑Shadow Functions Extended Capability Header <ins>↑↑Shadow Functions Extended Capability Header</ins> | 1496 |
| Table 7-327 | ↑↑Shadow Functions Capability Register <ins>↑↑Shadow Functions Capability Register</ins> | 1497 |
| Table 7-328 | ↑↑Shadow Functions Control Register <ins>↑↑Shadow Functions Control Register</ins> | 1497 |
| Table 7-329 | ↑↑Shadow Functions Instance Register Entry <ins>↑↑Shadow Functions Instance Register Entry</ins> .. | 1498 |
| Table 7-330 | IDE Extended Capability Header..... | 1499 |
| Table 7-331 | IDE Capability Register..... | 1500 |
| Table 7-332 | IDE Control Register | 1502 |
| Table 7-333 | Link IDE Stream Control Register | 1503 |
| Table 7-334 | Link IDE Stream Status Register | 1505 |
| Table 7-335 | Selective IDE Stream Capability Register | 1506 |
| Table 7-336 | Selective IDE Stream Control Register | 1506 |
| Table 7-337 | Selective IDE Stream Status Register | 1509 |
| Table 7-338 | IDE RID Association Register 1 (Offset +00h) | 1510 |
| Table 7-339 | IDE RID Association Register 2 (Offset +04h) | 1510 |
| Table 7-340 | IDE Address Association Register 1 (Offset +00h) | 1511 |
| Table 7-341 | IDE Address Association Register 2 (Offset +04h) | 1511 |
| Table 7-342 | IDE Address Association Register 3 (Offset +04h) | 1512 |
| Table 7-343 | Null Capability | 1512 |
| Table 7-344 | Null Extended Capability | 1513 |
| Table 7-345 | Streamlined Virtual Channel Extended Capability Header | 1514 |
| Table 7-346 | SVC Port Capability Register 1 | 1515 |
| Table 7-347 | SVC Port Control Register | 1516 |
| Table 7-348 | SVC Port Status Register..... | 1516 |
| Table 7-349 | SVC Resource Capability Register | 1517 |
| Table 7-350 | SVC Resource Control Register | 1518 |
| Table 7-351 | SVC Resource Status Register | 1519 |
| Table 7-352 | MRBL Extended Capability Header | 1521 |
| Table 7-353 | MRBL Capabilities Register | 1521 |
| Table 7-354 | MRBL Locator Register | 1522 |
| Table 8-1 | Tx Preset Ratios and Corresponding Coefficient Values for 8.0, 16.0, and 32.0 GT/s | 1529 |
| Table 8-2 | Tx Preset Ratios and Corresponding Coefficient Values for 64.0 GT/s | 1530 |
| Table 8-3 | Cases that the Reference Packages and ps21 _{TX} Parameter are Normative | 1537 |
| Table 8-4 | Recommended De-embedding Cutoff Frequency | 1548 |
| Table 8-5 | Tx Measurement and Post Processing For Different RefClks | 1550 |
| Table 8-6 | Data Rate Dependent Transmitter Parameters | 1558 |
| Table 8-7 | Data Rate Independent Tx Parameters | 1566 |
| Table 8-8 | Calibration Channel IL Limits | 1569 |
| Table 8-11 | Stressed Jitter Eye Parameters | 1590 |
| Table 8-12 | Common Receiver Parameters | 1602 |
| Table 8-13 | Lane Margining..... | 1606 |
| Table 8-14 | Package Model Capacitance Values | 1612 |
| Table 8-15 | Jitter/Voltage Parameters for Channel Tolerancing | 1624 |
| Table 8-16 | Channel Tolerancing Eye Mask Values | 1627 |
| Table 8-17 | EIEOS Signaling Parameters | 1630 |

Base 6.4 vs Base 6.3

| | | |
|-------------|--|--------------------|
| Table 8-18 | REFCLK DC Specifications and AC Timing Requirements | 1631 |
| Table 8-19 | Data Rate Independent Refclk Parameters | 1635 |
| Table 8-20 | Jitter Limits for CC Architecture | 1641 |
| Table 8-21 | Form Factor Clocking Architecture Requirements | 1641 |
| Table 8-22 | Form Factor Common Clock Architecture Details | 1642 |
| Table 8-23 | Form Factor Clocking Architecture Requirements Example | 1642 |
| Table 8-24 | Form Factor Common Clock Architecture Details Example | 1642 |
| Table 9-1 | VF Routing ID Algorithm | 1661 |
| Table 9-2 | SR-IOV Extended Capability Header | 1668 |
| Table 9-3 | SR-IOV Capabilities Register | 1668 |
| Table 9-4 | SR-IOV Control Register..... | 1671 |
| Table 9-5 | SR-IOV Status | 1675 |
| Table 9-8 | BAR Offsets | 1680 |
| Table 9-9 | SR-IOV Usage of PCI Standard Capabilities | 1681 |
| Table 9-10 | SR-IOV Usage of PCI Express Extended Capabilities | 1681 |
| Table 9-11 | ↑↑SIOV Extended Capability Header↑↑ | 1685 |
| Table 9-12 | ↑↑SIOV Capabilities Register↑↑ | 1686 |
| Table 9-13 | ↑↑SIOV Status Register↑↑ | 1687 |
| Table 10-1 | Address Type (AT) Field Encodings | 1698 |
| Table 10-2 | Translation Completion Status Codes..... | 1702 |
| Table 10-3 | Translation Completion Data Fields..... | 1704 |
| Table 10-5 | Examples of Translation Size Using S Field | 1706 |
| Table 10-6 | Page Request Message Data Fields..... | 1726 |
| Table 10-7 | PRG Response Message Data Fields | 1731 |
| Table 10-8 | Response Codes | 1731 |
| Table 10-9 | ATS Extended Capability Header | 1733 |
| Table 10-10 | ATS Capability Register (Offset 04h) | 1734 |
| Table 10-11 | ATS Control Register..... | 1735 |
| Table 10-13 | Page Request Extended Capability Header | 1737 |
| Table 10-14 | Page Request Control Register | 1737 |
| Table 10-15 | Page Request Status Register | 1738 |
| Table 11-1 | ↑↑XT Bit and T Bit Encodings↑↑ | 1746 |
| | | ECN: Base 6.3 XT△↔ |
| Table 11-2 | INTERFACE_ID Definition | 1749 |
| Table 11-3 | Example DSM Tracking and Handling for Architected Registers | 1757 |
| Table 11-4 | TDISP Request Codes | 1760 |
| Table 11-5 | TDISP Response Codes | 1761 |
| Table 11-6 | TDISP Message Format | 1762 |
| Table 11-7 | Generic Error Response Codes | 1763 |
| Table 11-8 | TDISP_VERSION..... | 1763 |
| Table 11-9 | GET_TDIP_CAPABILITIES..... | 1764 |
| Table 11-10 | TDISP_CAPABILITIES..... | 1764 |
| Table 11-11 | LOCK_INTERFACE_REQUEST | 1766 |
| Table 11-12 | LOCK_INTERFACE_RESPONSE | 1767 |
| Table 11-13 | LOCK_INTERFACE_REQUEST Error Codes | 1768 |
| Table 11-14 | GET_DEVICE_INTERFACE_REPORT | 1768 |
| Table 11-15 | DEVICE_INTERFACE_REPORT | 1769 |
| Table 11-16 | TDI Report Structure | 1769 |
| Table 11-17 | GET_DEVICE_INTERFACE_REPORT Error Response Codes | 1771 |
| Table 11-18 | DEVICE_INTERFACE_STATE | 1772 |
| Table 11-19 | START_INTERFACE_REQUEST | 1772 |
| Table 11-20 | START_INTERFACE_REQUEST Error Response Codes | 1772 |
| Table 11-21 | BIND_P2P_STREAM_REQUEST | 1774 |

Base 6.4 vs Base 6.3

| | | |
|-------------|---|------|
| Table 11-22 | BIND_P2P_STREAM_REQUEST Error Response Codes | 1774 |
| Table 11-23 | UNBIND_P2P_STREAM_REQUEST | 1775 |
| Table 11-24 | UNBIND_P2P_STREAM_REQUEST Error Response Codes | 1776 |
| Table 11-25 | SET_MMIO_ATTRIBUTE_REQUEST | 1777 |
| Table 11-26 | SET_MMIO_ATTRIBUTE_REQUEST Error Response Codes | 1777 |
| Table 11-27 | TDISP_ERROR | 1778 |
| Table 11-28 | Error Code and Error Data | 1778 |
| Table 11-29 | EXTENDED_ERROR_DATA | 1779 |
| Table 11-30 | VDM_REQUEST | 1779 |
| Table 11-31 | VDM_RESPONSE | 1779 |
| Table 11-32 | ↑↑SET_TDISP_CONFIG_REQUEST↑↑ | 1780 |
| Table 11-33 | ↑↑SET_TDISP_CONFIG_RESPONSE↑↑ | 1780 |
| Table 11-34 | Example TSM Tracking and Handling for Root Port Configurations | 1790 |
| Table 12-1 | Relative Comparisons of Typical Architectural Out-of-Band Interfaces | 1797 |
| Table 12-2 | PESTI DC Specifications | 1805 |
| Table 12-3 | PESTI Initiator Control and Status Registers | 1808 |
| Table 12-4 | PESTI Discovery Status State Transitions | 1808 |
| Table 12-5 | PESTI AC Specifications | 1810 |
| Table 12-6 | PESTI Discovery Payload | 1812 |
| Table 12-7 | Example VWIRE_OUT_0 (Initiator to Target) | 1815 |
| Table 12-8 | Example VWIRE_IN_0 (Target to Initiator) | 1815 |
| Table 12-9 | MSC_CTRL_VAL (Initiator to PESTI Snooper Target) Data Byte Value = 02h | 1818 |
| Table 12-10 | MSC_STAT_VAL (PESTI Snooper Target to Initiator) | 1819 |
| Table 12-11 | ↑↓2-wire↓↑↑2-Wire↑↑ Interface Example Usages | 1822 |
| Table 12-12 | Baseline SMBus Recommended Default Target Addresses | 1823 |
| Table 12-13 | I3C Basic Logic Signaling DC Specification | 1830 |
| Table 12-14 | I3C Timing Requirements | 1830 |
| Table 12-15 | PCI-SIG MultiRecord | 1835 |
| Table 12-17 | PCI-SIG MultiRecord Descriptor | 1836 |
| Table 12-18 | Descriptor Sub-Types for Group ID 0h | 1837 |
| Table 12-19 | Connector Subdivision Combinations Descriptor | 1838 |
| Table 12-20 | ↑↓Connector Subdivision Descriptor↓↑↑Connector Subdivision Descriptor↑↑ | 1838 |
| Table 12-21 | ↑↑PCIe-MI Command↑↑ | 1852 |
| Table 12-24 | ↑↑PCIe-MI Completion↑↑ | 1853 |
| Table 12-27 | ↑↑PCIe-MI Opcodes↑↑ | 1854 |
| Table 12-28 | ↑↑PCIe-MI Status Codes↑↑ | 1855 |
| Table 12-29 | ↑↑Invalid Input Error Completion↑↑ | 1856 |
| Table 12-31 | ↑↑Information Identifiers↑↑ | 1859 |
| Table 12-32 | ↑↑Get Information Command↑↑ | 1860 |
| Table 12-33 | ↑↑Get Information – Supported Information Identifiers Completion↑↑ | 1860 |
| Table 12-34 | ↑↑Get Information – Supported Opcodes Completion↑↑ | 1861 |
| Table 12-35 | ↑↑Get Information – Supported Parameters Completion↑↑ | 1861 |
| Table 12-39 | ↑↑Get Information – Component Information Completion↑↑ | 1862 |
| Table 12-40 | ↑↑Get Information – Port Information Command↑↑ | 1863 |
| Table 12-41 | ↑↑Get Information – Port Information Completion↑↑ | 1863 |
| Table 12-44 | ↑↑Get Information – PCIe Port-Specific Information Completion↑↑ | 1864 |
| Table 12-47 | ↑↑Get Information – Function List Command↑↑ | 1866 |
| Table 12-49 | ↑↑Get Information – Function List Completion↑↑ | 1867 |
| Table 12-50 | ↑↑Function List Data Structure↑↑ | 1868 |
| Table 12-51 | ↑↑Get Information – Function Information Command↑↑ | 1868 |
| Table 12-52 | ↑↑Get Information – Function Information Completion↑↑ | 1868 |
| Table 12-55 | ↑↑Capability Structure Data Structure↑↑ | 1870 |

Base 6.4 vs Base 6.3

| | | |
|-------------|--|------|
| Table 12-56 | ↑↑Parameter Identifiers↑..... | 1870 |
| Table 12-57 | ↑↑Get Parameters Command↑..... | 1871 |
| Table 12-58 | ↑↑Set Parameters Command↑..... | 1871 |
| Table 12-59 | ↑↑Get Parameters – Configuration Space Writability Command↑..... | 1874 |
| Table 12-60 | ↑↑Get Parameters – Configuration Space Writability Completion↑..... | 1874 |
| Table 12-61 | ↑↑Configuration Space Writability Data Structure↑..... | 1875 |
| Table 12-66 | ↑↑Set Parameters – Configuration Space Writability Command↑..... | 1876 |
| Table 12-71 | ↑↑Get Parameters – Routing Information Command↑..... | 1879 |
| Table 12-72 | ↑↑Get Parameters – Routing Information Completion↑..... | 1879 |
| Table 12-73 | ↑↑Set Parameters – Routing Information Command↑..... | 1880 |
| Table 12-74 | ↑↑Get Parameters – Configuration Space Command↑..... | 1881 |
| Table 12-75 | ↑↑Get Parameters – Configuration Space Completion↑..... | 1881 |
| Table 12-76 | ↑↑Set Parameters – Configuration Space Command↑..... | 1882 |
| Table 12-77 | ↑↑Get Parameters – Connector Subdivision Command↑..... | 1882 |
| Table 12-78 | ↑↑Get Parameters – Connector Subdivision Completion↑..... | 1883 |
| Table 12-79 | ↑↑Set Parameters – Connector Subdivision Command↑..... | 1883 |
| Table 12-80 | ↑↑Set Parameters – Connector Subdivision Completion↑..... | 1884 |
| Table A-1 | Isochronous Bandwidth Ranges and Granularities | 1890 |
| Table B-1 | 8b/10b Data Symbol Codes | 1897 |
| Table B-2 | 8b/10b Special Character Symbol Codes..... | 1905 |
| Table F-1 | Message Code Usage..... | 1925 |
| Table F-2 | PCI-SIG-Defined VDM Subtype Usage..... | 1926 |
| Table G-1 | PCI Express Attribute Impact on Protocol Multiplexing..... | 1929 |
| Table G-2 | PMUX Attribute Impact on PCI Express | 1931 |
| Table G-3 | PMUX Packet Layout (8b/10b Encoding)..... | 1933 |
| Table G-4 | PMUX Packet Layout (128b/130b Encoding)..... | 1935 |
| Table G-5 | Symbol 1 Bits [6:3]..... | 1936 |
| Table G-6 | PMUX Extended Capability Header..... | 1938 |
| Table G-7 | PMUX Capability Register..... | 1939 |
| Table G-8 | PMUX Control Register | 1940 |
| Table G-9 | PMUX Status Register | 1941 |
| Table G-10 | PMUX Protocol Array Entry | 1943 |
| Table H-1 | Maximum UpdateFC Transmission Latency Guidelines for 2.5 GT/s Mode Operation by Link Width and Max Payload (Symbol Times)..... | 1946 |
| Table H-2 | Maximum UpdateFC Transmission Latency Guidelines for 5.0 GT/s Mode Operation by Link Width and Max Payload (Symbol Times)..... | 1946 |
| Table H-3 | Maximum UpdateFC Transmission Latency Guidelines for 8.0 GT/s Operation by Link Width and Max Payload (Symbol Times) | 1947 |
| Table H-5 | Maximum Ack Latency Limit and AckFactor for 2.5 GT/s (Symbol Times) | 1948 |
| Table H-6 | Maximum Ack Transmission Latency Limit and AckFactor for 5.0 GT/s (Symbol Times) | 1949 |
| Table H-7 | Maximum Ack Transmission Latency Limit and AckFactor for 8.0 GT/s (Symbol Times) | 1949 |
| Table L-1 | Inputs and Outputs for Example IDE TLP | 2257 |
| Table L-2 | Inputs and Outputs for Example IDE TLP (FM) | 2261 |
| Table L-3 | Inputs and Outputs for Example IDE TLP with Partial Header Encryption (mode 0100b - Address[41:2] Encrypted) (NFM) | 2264 |
| Table L-4 | Inputs and Outputs for Example IDE TLP with Partial Header Encryption (mode 0100b - Address[41:2] Encrypted) (FM) | 2268 |
| Table L-5 | IDE Test Keys..... | 2270 |

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current PCISIG publications and the latest revision of this specification can be found at pcisig.com

This is the PCI Express Base [¶↓6.3↑](#) [¶↑6.4↑](#) Specification . This consists of Base [¶↓6.2↓](#) [¶↑6.3↑](#) plus approved ECNs, errata, and editorial corrections. See [Critical Errata](#) and [Important Errata](#) and [Approved ECNs](#).

- The [¶↓NCB-PCI_Express_Base_6.3.pdf↓](#) [¶↑NCB-PCI_Express_Base_6.4.pdf↑](#) is normative (i.e., the official specification). It contains no changebars.
- The [¶↓CB-PCI_Express_Base_6.3-vs-6.2.pdf↓](#) [¶↑CB-PCI_Express_Base_6.4-vs-6.3.pdf↑](#) is informative. It contains changebars relative to the PCI Express Base [¶↓6.2↓](#) [¶↑6.3↑](#) Specification
[¶↓\[PCIe-6.2\].↓](#) [¶↑\[PCIe-6.3\].↑](#)

NOTE: Background on the new Document Process §

The new PCISIG document system is a variant of the w3c Respec tool (see <https://github.com/w3c/respec/wiki>). Respec is a widely used tool written to support the World Wide Web specifications. The PCISIG variant is <https://github.com/sglaser/respec> . Both Respec and the PCISIG variant are open source (MIT License) Javascript libraries. They operate in the author's browser and provide a rapid edit / review cycle without requiring any special tools be installed.

Respec is built on top of HTML5, the document format for the World Wide Web <http://www.w3.org/TR/html5/> . HTML is a text-based document format that allows us to deploy tools commonly used for software development (git, continuous integration, build scripts, etc.) to better manage and control the spec development process.

PCISIG enhancements to Respec support document formatting closer to existing PCISIG practice as well as automatic creation of register figures (eliminating about half of the manually drawn figures).

NOTE: Navigating in changebar documents §

The Base 6.0.1 version introduces a new errata delivery process. Instead of having a separate “change this to that” document, a new version of the specification is produced with the changes integrated into the document. This makes it easier to consume and less likely that errata will be overlooked.

All changes are annotated with the **↑** character. Searching for this character in a PDF reader will step through all changes. There is a **↑** character in the upper right of page 1 that can be copied into the search string.

Inserted text is yellow and underlined. It contains the **↑** character.

Deleted text is red and struck through. It contains the **↓** character.

All errata are identified by a “red box” near the first change in a chunk of changes. This box contains one or more triangle characters that can navigate through the changes for the associated errata.

- △ The upward pointing triangle is a link to the associated errata table entry.
- ▷ The right pointing triangle is a link to the next change associated with this errata. If this is the last change, this triangle is not present.
- ◁ The left pointing triangle is a link to the previous change associated with this errata. If this is the first
↓↓chance,↑ ↑↑change,↑ this triangle is not present.

Changes that are not marked with a red box are considered editorial in nature and are not associated with an errata.

Where the automated process can't correctly identify changes (e.g., figures and equations), the **↓↓errata table↓**
↑↑revision history↑ contains the change description. **↑↑When fields are added or removed from registers, the automatically drawn figure before the register table may not show these fields. The table will correctly show such fields as will the drawings in the non-changebar version of the document. Since the table accurately shows the change, the revision history doesn't call out the "missing" figure change.↑**

Publication does not imply endorsement by the PCI-SIG Membership. This document may be updated, replaced or obsoleted by other documents at any time.

This document is governed by the PCI-SIG Specification Development Procedures.

Base 6.4 vs Base 6.3

Revision History

| Revision | Revision History | Date |
|----------|--|------------|
| 1.0 | Initial release. | 07/22/2002 |
| 1.0a | Incorporated Errata C1-C66 and E1-E4.17. | 04/15/2003 |
| 1.1 | Incorporated approved Errata and ECNs. | 03/28/2005 |
| 2.0 | Added 5.0 GT/s data rate and incorporated approved Errata and ECNs. | 12/20/2006 |
| 2.1 | <p>Incorporated <i>Errata for the PCI Express Base Specification, Rev. 2.0</i> (February 27, 2009), and added the following ECNs:</p> <ul style="list-style-type: none"> Internal Error Reporting ECN (April 24, 2008) Multicast ECN (December 14, 2007, approved by PWG May 8, 2008) Atomic Operations ECN (January 15, 2008, approved by PWG April 17, 2008) Resizable BAR Capability ECN (January 22, 2008, updated and approved by PWG April 24, 2008) Dynamic Power Allocation ECN (May 24, 2008) ID-Based Ordering ECN (January 16, 2008, updated 29 May 2008) Latency Tolerance Reporting ECN (22 January 2008, updated 14 August 2008) Alternative Routing-ID Interpretation (ARI) ECN (August 7, 2006, last updated June 4, 2007) Extended Tag Enable Default ECN (September 5, 2008) TLP Processing Hints ECN (September 11, 2008) TLP Prefix ECN (December 15, 2008) | 03/04/2009 |
| 3.0 | <p>Added 8.0 GT/s data rate, latest <ins>incorporated</ins> approved Errata, and the following ECNs:</p> <ul style="list-style-type: none"> Optimized Buffer Flush/Fill ECN (8 February 2008, updated 30 April 2009) ASPM Optionality ECN (June 19, 2009, approved by the PWG August 20, 2009) Incorporated End-End TLP Changes for RCs ECN (26 May 2010) and Protocol Multiplexing ECN (17 June 2010) | 11/10/2010 |
| 3.1 | <p>Incorporated feedback from Member Review</p> <p>Incorporated Errata for the PCI Express® Base Specification Revision 3.0</p> <p>Incorporated M-PCIe Errata (3p1_active_errata_list_mpcie_28Aug2014.doc and 3p1_active_errata_list_mpcie_part2_11Sept2014.doc)</p> <p>Incorporated the following ECNs:</p> <ul style="list-style-type: none"> ECN: Downstream Port containment <ins>Containment</ins> (DPC) ECN: Separate Refclk Independent SSC (SRIS) <ins>Architecture</ins> (SRIS) ECN: Process Address Space ID (PASID) | 10/8/2014 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|--|------------|
| | <ul style="list-style-type: none"> • ECN: Lightweight Notification (LN) Protocol • ECN: Precision Time Measurement • ECN: Enhanced DPC (eDPC) • ECN: 8.0 GT/s Receiver Impedance • ECN: L1 PM Substates with CLKREQ • ECN: Change Root Complex Event Collector Class Code • ECN: M-PCIe • ECN: Readiness Notifications (RN) • ECN: Separate Refclk Independent SSC Architecture (SRIS) JTOL and SSC Profile Requirements | |
| 3.1a | <p>Minor update:</p> <p>Corrected: Equation 4.3.9 in Section 4.3.8.5., Separate Refclk With Independent SSC (SRIS) Architecture. Added missing square (exponent=2) in the definition of B.</p> <p>$B = 2.2 \times 10^{12} \times (2\pi)^2$ where \wedge = exponent.</p> | 12/5/2015 |
| 4.0 | <p>Version 0.3: Based on PCI Express® Base Specification Revision 3.1 (October 8, 2014) with some editorial feedback received in December 2013.</p> <ul style="list-style-type: none"> • Added § Chapter 9.1, Electrical Sub-block: Added § Chapter 9.1 (Rev0.3-11-30-13_final.docx) • Changes related to Revision 0.3 release • Incorporated PCIe-relevant material from PCI Bus Power Management Interface Specification (Revision 1.2, dated March 3, 2004). This initial integration of the material will be updated as necessary and will supersede the standalone Power Management Interface specification. <p>Version 0.5 (12/22/14, minor revisions on 1/26/15, minor corrections 2/6/15)</p> <ul style="list-style-type: none"> • Added front matter with notes on expected discussions and changes. • Added ECN:Retimer (dated October 6, 2014) • Corrected § Chapter 4. title to, “Physical Layer Logical Block”. • Added Encoding subteam feedback on § Chapter 4. • Added Electrical work group changes from PCIe Electrical Specification Rev 0.5 RC1 into § Chapter 9.1 | 2/6/2015 |
| | <p>Version 0.7: Based on PCI Express® Base Specification Version 4.0 Revision 0.5 (11/23/2015)</p> <ul style="list-style-type: none"> • Added ECN_DVSEC-2015-08-04 • Applied ECN PASID-ATS dated 2011-03-31 • Applied PCIE Base Spec Errata: PCIe_Base_r3_1_Errata_2015-09-18 except: <ul style="list-style-type: none"> ◦ B216; RCIE ◦ B256; grammar is not clear • Changes to Chapter 7. Software Initialization and Configuration per PCIe_4.0_regs_0-3F_gord_7.docx | 11/24/2015 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|---|--|-----------------------|
| | <ul style="list-style-type: none"> Added Chapter SR-IOV Spec Rev 1.2 (Rev 1.1 dated September 8, 2009 plus: <ul style="list-style-type: none"> SR-IOV_11_errata_table.doc DVSEC 3.1 Base Spec errata Added Chapter ATS Spec Rev 1.2 (Rev 1.1 dated January 26, 2009 plus: <ul style="list-style-type: none"> ECN-PASID-ATS 3.1 Base Spec errata | |
| 2/18/2016 Changes from the Protocol Working Group | <ul style="list-style-type: none"> Applied changes from the following documents: <ul style="list-style-type: none"> FC Init/Revision scaled-flow-control-pcie-base40-2016-01-07.pdf (Steve.G) Register updates for integrated legacy specs PCIe_4.0_regs_0-3F_gord_8.docx (GordC) Tag Scaling PCIe 4_0 Tag Field scaling 2015-11-23 clean.docx (JoeC) MSI/MSI-X PCIe 4_0 MSI & MSI-X 2015-12-18 clean.docx (JoeC); register diagrams TBD on next draft. REPLAY_TIMER/Ack/FC Limits Ack_FC_Replay_Timers_ver8 (PeterJ) | 2/18/16 |
| Chapter 10. SR-IOV related changes: | <ul style="list-style-type: none"> Incorporated “SR-IOV and Sharing Specification” Revision 1.1 dated January 20, 2010 (sr-iov1_1_20Jan10.pdf) as § Chapter 10., with changes from the following documents <ul style="list-style-type: none"> Errata for the PCI Express® Base Specification Revision 3.1, Single Root I/O Virtualization and Sharing Revision 1.1, Address Translation and Sharing Revision 1.1, and M.2 Specification Revision 1.0: PCIe_Base_r3_1_Errata_2015-09-18_clean.pdf ECN__Integrated_Endpoints_and_IOV_updates__19 Nov 2015_Final.pdf Changes marked “editorial” only in marked PDF: sr-iov1_1_20Jan10-steve-manning-comments.pdf | 4/26/16 [snapshot] |
| Chapter 9. Electrical Sub-Block related changes: Source: WG approved word document from Dan Froelich (FileName: Electrical-PCI_Express_Base_4.0r0.7_April_7_wg_approved_redo_for_figure_corruption.docx) | | 5/23/ 16[snapshot] |
| Version 0.7 continued... Chapter 4. PHY Logical Changes based on: <ul style="list-style-type: none"> Chapter4-PCI_Express_Base_4_0r0_7_May3_2016_draft.docx Chapter 7.. PHY Logical Changes based on: <ul style="list-style-type: none"> PCI_Express_Base_4_0r0_7_Physical-Logical_Ch7_Delta_28_Apr_2016.docx | | |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|---|---------|
| | <p>----- Changes incorporated into the August 2016 4.0 r0.7 Draft PDF -----</p> <p>June 16 Feedback from PWG on the May 2016 snapshot</p> <p>PWG Feedback on 4.0 r0.7 Feb-Apr-May-2016 Drafts</p> <p>*EWG Feedback:</p> <ul style="list-style-type: none"> -CB-PCI_Express_Base_4.0r0.7_May-2016 (Final).fdf <p>-EWG f/b:</p> <ul style="list-style-type: none"> Electrical-PCI_Express_Base_4.0r0.7_April_7_wg_approved_redo_for_figure_corruption_Broadco.docx <p>*PWG Feedback:</p> <ul style="list-style-type: none"> -PWG 0.7 fix list part1 and part 2.docx -PWG 0 7 fix list part3a.docx -PCI_Express_Base_4.0r0.7_pref_April-2016_chp5_PM_stuff_only_ver3.docx -PCI_Express_Base_4.0r0.7_pref_April-2016_chp5_PM_stuff_only_ver3.docx -scaled-flow-control-pcie-base40-2016-07-07.pdf -ECN_NOP_DLLP-2014-06-11_clean.pdf -ECN_RN_29_Aug_2013.pdf -3p1_active_errata_list_mpcie_28Aug2014.doc -3p1_active_errata_list_mpcie_part2_11Sept2014.doc -lane-margining-capability-snapshot-2016-06-16.pdf -Emergency Power Reduction Mechanism with PWRBRK Signal ECN -PWG 0 7 fix list part4.docx -ECN_Conventional_Adv_Caps_27Jul06.pdf -10-bit Tag related SR-IOV Updates <p>*Other:</p> <ul style="list-style-type: none"> -Merged Acknowledgements back pages from SR-IOV and ATS specifications into the main base spec. Acknowledgements page. | 8/30/16 |
| | <p>----- Changes since August 2016 for the September 2016 4.0 r0.7 Draft PDF-----</p> <p>Applied:</p> <p>PWG Feedback/Corrections on August draft</p> <p>ECN_SR-IOV_Table_Updates_16-June-2016.doc</p> | 9/28/16 |
| | <p>----- Changes since September 28 2016 for the October 2016 4.0 r0.7 Draft PDF-----</p> <p>EWG:</p> <p>Updates to § Chapter 9.1 - Electrical Sub-block (Sections: 9.4.1.4, 9.6.5.1, 9.6.5.2, 9.6.7)</p> <p>PWG:</p> <p>Updates to Sections: 3.2.1, 3.3, 3.5.1, 7.13, 7.13.3 (Figure: Data Link Status Register)</p> | 10/7/16 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|---|---------------|
| | <p>----- Changes to the October 13 2016 4.0 r0.7 Draft PDF -----</p> <p>EWG: Updates to <u>¶§ Chapter 9.1</u> - Electrical Sub-block (<u>§ Section 9.4.3.9</u> and Figure 9-9 caption)</p> | 10/21/16 |
| | <p>----- Changes to the November 3 2016 4.0 r0.7 Draft PDF -----</p> <p><u>§ Section 2.6.1 Flow Control Rules:</u> Updated Scaled Flow Control sub-bullet under FC initialization bullet (before Table 2-43)</p> | 11/3/16 |
| | <p>----- Changes to the November 11 2016 4.0 r0.7 Draft PDF -----</p> <p>Added M-PCIe statement to the Open Issues page Updated date to November 11, 2016</p> | 11/11/16 |
| | <p>-----</p> <p>Version 0.9: Based on PCI Express® Base Specification Version 4.0 Revision 0.7 (11/11/2016) Incorporated the following ECNs:</p> <ul style="list-style-type: none"> -ECN-Hierarchy_ID-2017-02-23 -ECN_FPB_9_Feb_2017 -ECN Expanded Resizable BARs 2016-04-18 -ECN-VF-Resizable-BARs_6-July-2016 - <u>§ Chapter 7.</u> reorganized: <ul style="list-style-type: none"> • New section 7.6 created per a PWG-approved reorganization to move sections 7.5, 7.6., and 7.10 to subsections 7.6.1 through 7.6.3 resp. • New section 7.7 created per a PWG-approved reorganization to move sections 7.7, 7.8., 7.12, 7.13, 7.40, 7.41 and 7.20 to subsections 7.7.1 through 7.7.7 resp. • New section 7.9 created per a PWG-approved reorganization to move sections 7.15, 7.22, 7.16, 7.23, 7.39, 7.24, 7.17, 7.18, 7.21, 7.25, 7.28, 7.30, 7.33, 7.34, 7.35, 7.38, and 7.42 to subsections 7.9.1 through 7.9.17 resp. <p>-Removed <u>§ Chapter 8.</u> : M-PCIe Logical Sub-Block</p> <p>-Updated <u>¶§ Chapter 9.1</u> (8 now), EWG Updates to <u>¶§ Chapter 9.1</u> - Electrical Sub-block per: Chapter9-PCI_Express_Base_4 0r09_March_30-2017_approved.docx</p> <p>-Updated <u>§ Chapter 4.</u> : Physical Layer Logical Block per PCI_Express_Base_4 0_r0 9_Chapter4_Final_Draft.docx</p> <p>-Updated Figures in <u>§ Chapter 10.</u> : ATS Specification</p> <p>-Removed <u>§ Appendix H.</u> : M-PCIe timing Diagrams</p> <p>-Removed Appendix I: M-PCIe Compliance Patterns, pursuant to removing the M-PCIe Chapter this 0.9 version of the 4.0 Base Spec.</p> <p>-Added <u>§ Appendix H.</u> : Flow Control Update Latency and ACK Update Latency Calculations</p> <p>-Added Appendix I: Vital Product Data (VPD)</p> | April 28 2017 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|--------------|---|-----------------|
| | <ul style="list-style-type: none"> -Updated editorial feedback on the Appendix section per: PCI_Express_Base_4.0r0.7_appendices_November-11-2016_combined-editorial.docx -Deleted references to M-PCIe throughout the document. -Updated <u>§ Chapter 9.1</u> (8 now), EWG Updates to <u>§ Chapter 9.1</u> - Electrical Sub-block per: Chapter9-PCI_Express_Base_4_0r09_March_30-2017_approved.docx -Updated <u>§ Chapter 4</u> : Physical Layer Logical Block per PCI_Express_Base_4_0_r0 9_Chapter4_Final_Draft.docx -Updated Figures in <u>§ Chapter 10</u> : ATS Specification -Added <u>§ Appendix H</u> : Flow Control Update Latency and ACK Update Latency Calculations -Following items that were marked deleted in the Change Bar version of the April 28th snapshot have been “accepted” to no longer show up: pp 1070: Lane Equalization Control 2 Register (Offset TBD) Comment: Deleted per: PCI_Express_Base_4_0r0_7_Physical-Logical_Ch7_Delta_28_Apr_2016.docx pp 1074: Physical Layer 16.0 GT/s Margining Extended Capability section Comment: Deleted per: PCI_Express_Base_4_0r0_7_Physical-Logical_Ch7_Delta_28_Apr_2016.docx Comment: Replaced by Section Lane Margining at the Receiver Extended Capability per Fix3a #83 <u>lane-margining-capability-snapshot-2016-06-16.pdf</u> -Incorporated: PCIe 4_0 Tag Field scaling 2017-03-31.docx -Vital Product Data (VPD) -Added <u>§ Section 6.27</u> -Added <u>§ Section 7.9.4</u> -Incorporated feedback from April 28th snapshot.[source: 3 fdf files] -Completed editorial feedback on the Appendix section per: PCI_Express_Base_4.0r0.7_appendices_November-11-2016_combined-editorial.docx -Incorporated ECN EMD for MSI 2016-05-10 -Updated per: PWG F2F changes from: PCI_Express_Base_4.0r0.7_pref_November-11-2016-F2F-2017-03-16-2017-03-30-sdg.docx -Updated figures per following lists (Gord Caruk): PCIe_4_0_fix_drawing_items.doc PCIe_4_0_fix_drawing_items_part2.doc | May 26, 2017 |
| Version 0.91 | <p>***Note this version will be used as the base for the PCI Express® Base Specification Revision 5.0***</p> <p>Item numbers are with reference to PWG CheckList (https://members.pcisig.com/wg/PCIe-Protocol/document/10642)</p> <ul style="list-style-type: none"> -Moved Flattening Portal Bridge Section 7.10 to Section 7.8.10. PWG Checklist Items #12.1 -Fixed misc. feedback that needed clarification from the 0.9 version. Issues fall under the categories of figure updates, broken cross references. Also incorporated feedback received from member review of the 4.0 version rev. 0.9 Base Spec. -Updated to reconcile issues related to incorporating the Extended Message Data for MSI ECN. PWG Checklist Items #22 -Completed incorporating all resolved editorial items from PWG Checklist Items #14, 14.1, 15.1, 36, 42. TBD: Some minor editorial items from #13, #14 and #15 have been deferred to post 0.91 by reviewers. TBD: Errata and NPEM ECN | August 17, 2017 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|--|--------------------|
| | <p>ECN: ECN_Native_PCIE_Enclosure_Management_v10August2017.docx</p> <p>Deleted Section 5.11.1 through Section 5.14</p> <p>Changes tracked by items 34.01 34.02 34.04 34.05 34.11 in the PWG checklist</p> <p>Errata: B265, C266, 267, 268, B269, A270, A271, B274, C275, B276, B277, B278, B279, B280, B281, B283, B284, B285, B286, B288, B289, B292, B293, B294, B295, B297, B299, B300, B301</p> <p>Other minor edits per: NCB-PCI_Express_Base_4.0r0.91_August-17-2017_dh_sdg_Annot_2.fdf</p> | August 28, 2017 |
| | <p>Applied fixes and corrections captured in NCB-PCI_Express_Base_4.0r1.0_August-28-2017.fdf (Revision 8):</p> <p>https://members.pcisig.com/wg/PCIe-Protocol/document/10770</p> <p>Updated contributor list in Appendix section.</p> | September 20, 2017 |
| | <p>Updated contributor list in Appendix section.</p> <p>Inserted correct Figure 6-2.</p> <p>Applied minor fixes and corrections captured in:</p> <p>NCB-PCI_Express_Base_4.0r1.0_September-20-2017 https://members.pcisig.com/wg/PCIe-Protocol/document/10770</p> | September 27, 2017 |
| | <p>“-c” version: Changes to match -b version of the Final NCB PDF approved by PWG and EWG on September 29, 2017. See change bars. Details include:</p> <p>EWG Changes:</p> <ul style="list-style-type: none"> -Typo in Equation 8-3; changed 1.6.0 GT/s to 16.0 GT/s - <u>§ Section 8.4.2.1</u>; corrected references from Table 8-11 to Table 8-10 - <u>§ Section 8.5.1.3.3 & § Section 8.5.1.4.3</u> (Figure 8-47); changed “median” to “mean” <p>PWG Changes:</p> <ul style="list-style-type: none"> -Sub-Sub-Bullet before Figure 4-27. Added “or higher” after 8.0 GT/s - <u>§ Section 5.12 Power Management Events</u>; deleted last two paragraphs and Implementation Note. -Updated Acknowledgements section with additional contacts. | September 29, 2017 |
| 5.0 | <p>Version 0.3</p> <p>Summary of intended changes for 5.0. This was a short document, referencing the PCI Express Base Specification but not including it.</p> | 2017-06-01 |
| 5.0 | <p>Version 0.5</p> <p>Further details on intended changes for 5.0. This was a short document, referencing the PCI Express Base Specification but not including it.</p> | 2017-11-02 |
| | <p>Version 0.7</p> | 2018-06-07 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|---|------------|
| | <p>This was the first release of Base 5.0 based on the 4.0 Specification text. The 4.0 specification was converted into HTML format during this process. This conversion process was imperfect but does not impact the new 5.0 material.</p> | |
| | <p>Version 0.9</p> <p>This includes:</p> <ul style="list-style-type: none"> Additional details regarding operating at 32.0 GT/s Corrections to match published Base 4.0 Redrawing of some figures PCIe_Base_r4_0_Errata_2018-10-04a.pdf ECN-Thermal-Reporting 2017May18.pdf ECN-Link-Activation-07-Dec-2017.pdf | 2018-10-18 |
| | <p>Version 1.0</p> <p>This includes:</p> <ul style="list-style-type: none"> Corrections and clarification for support of the 32.0 GT/s operation Editorial Changes: <ul style="list-style-type: none"> Rewrite misleading / confusing text Update terminology for consistency and accuracy Update grammar for readability Add many hotlinks / cross references Implement all 4.0 Errata Incorporate Expansion ROM Validation ECN Expansion ROM Validation ECN.pdf Incorporate Enhanced PCIe Precision Time Measurement (ePTM) ECN ECN_ePTM_10_January_2019.pdf Incorporate Root Complex Event Collector Bus Number Association ECN ECN_EventCollector_13Sept2018a.pdf Incorporate PCIe Link Activation ECN ECN_Link_Activation_07_Dec_2017.pdf Incorporate Advanced Capabilities for Conventional PCI ECN (updated for PCIe) ECN_Conventional_Adv_Caps_27Jul06.pdf Incorporate Async Hot-Plug Updates ECN ECN_Async_Hot-Plug_Updates_2018-11-29.pdf Incorporate ACS Enhanced Capability ECN ECN_ACS_25_Apr_2019_Clean.pdf Incorporate the Subsystem ID and Subsystem Vendor ID Capability, from the PCI-to-PCI Bridge Architecture Specification, Revision 1.2 (updated for PCIe) ppb12.pdf | 2019-05-16 |
| 6.0 | <p>Version 0.3</p> <p>Initial Release of Base 6.0, Standalone Document</p> | 2019-10-04 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|--|--|
| | <p>Version 0.5, Standalone Document</p> <ul style="list-style-type: none"> • Add L0p • Add Shared Flow Control • Update Physical Layer / Logical Sublayer ↑↓material↓ ↑↑material↑ • Add Deprecation items: <ul style="list-style-type: none"> ◦ MR-IOV ◦ Lightweight Notification (LN) • Update new TLP Header material • Update Electrical Layer material | 2020-01-30 |
| | <p>Version 0.7, Integrated Document</p> <p>First version relative to Base 5.0 Specification text.</p> <ul style="list-style-type: none"> • Incorporate Base 5.0 Errata Matching Errata document to be published. • Incorporate Approved Base 5.0 ECNs: <ul style="list-style-type: none"> ◦ ACS Enhanced Capability ◦ ATS Memory Attributes Shadow Functions ◦ CMA ◦ DOE ◦ PTM Byte Order Adaptation ◦ DMWr ◦ PASID for Untranslated Addr • Support Flit Mode changes in Chapter 2 <ul style="list-style-type: none"> ◦ 14 bit Tag support ◦ Flit Mode TLP Format changes, including translation rules • Support Shared Flow Control, L0p, NULL2 DLLP in Chapter 3 • Integrate Flit Mode material into Chapter 4 • Flit Mode changes to Error Handling in Chapter 6 <ul style="list-style-type: none"> ◦ TLP Translation Blocked error • Integrate PAM4 Electrical changes into Chapter 8 • Refactor SR-IOV Registers from Chapter 9 into Chapter 7 Moves “VFs do things this way” material next to the original. • Define Shared Flow Control Supported and Shared Flow Control Enable bits in Chapter 7 (removed in Version 0.9) <ul style="list-style-type: none"> ◦ <u>Virtual Channel Extended Capability</u> ◦ <u>Multi-Function Virtual Channel Extended Capability</u> • Define new capabilities associated with 64.0 GT/s and Flit Mode in Chapter 7 <ul style="list-style-type: none"> ◦ <u>Physical Layer 64.0 GT/s Extended Capability</u> ◦ <u>Flit Logging Extended Capability</u> ◦ <u>Device 3 Extended Capability Structure</u> | ↑↑2020-10-08↑ |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|---|------------|
| | <ul style="list-style-type: none"> ◦ Flit Performance Measurement Extended Capability ◦ Flit Error Injection Extended Capability • Deprecate material: <ul style="list-style-type: none"> ◦ MR-IOV ◦ Lightweight Notification (LN) | |
| | <p>Version 0.71, Integrated Document</p> <ul style="list-style-type: none"> • Incorporate Combined Power ECN • Incorporate IDE ECN, plus Partial Header Encryption • Incorporate Errata: B90a, B90b, B90c, B90d, B90g, B90h, B90j, B90k, B90l, B90m, B90n, B90o, B90w, B90x, B90y, B90z, B113, B114 • L0p updates • Shared Flow control Updates • PAM4, 64.0 GT/s electrical updates • Physical Layer, Logical Sub-block updates <ul style="list-style-type: none"> ◦ Rework Flit Ack/Nak protocol ◦ Training Set changes • Max Payload Supported changes • TLP Layout changes • 14 bit tag updates • SR-IOV updates • Data Link Feature DLLP updates • DRS is <i>MUST@FLIT</i>, explain how software should use • Specify additional error behavior | 2021-06-24 |
| | <p>Version 0.9, Integrated Document</p> <ul style="list-style-type: none"> • Implement Errata B23, B64, B66-67, B74-75, B77, B78, B80-81, B85-88, B89a, B90f, B93-B95, B98-101, B103-105, B107-111, B112, B115-118, B120-121, B123-125, B127-128, B132, B133a-B133c, B133e • Simplified Shared Flow Control (always on in Flit Mode) and improvements to text • Numerous Phy Logical updates and issue fixes • Updated and Improved Flit Mode TLP type earmarking • Updated and Improved OHC content • Improve Segment-related content • Improve Max Payload Size content • <i>MUST@FLIT</i> changes, esp especially for Completion Timeout mechanism • Numerous editorial improvements | 2021-09-23 |
| | <p>Version 1.0, Published Document</p> <ul style="list-style-type: none"> • Incorporate Relaxed Detect Timing ECN | 2021-12-16 |

Base 6.4 vs Base 6.3

| Revision | Revision History | Date |
|----------|--|------|
| | <ul style="list-style-type: none"> Numerous editorial improvements Update errata B117 Add missing artwork Fix and add cross references Electrical section clarifications Add Remote L0p Supported bit in Device Status 3 Define Null Capability and Null Extended Capability Update Reference Documents Update encoding of Flit Mode Local TLP Prefix (was inconsistent in 0.9) | |

Critical Errata

| Revision | Errata | Description |
|----------|--------|--|
| 6.0.1 | A20 | <p>Partial Header Encryption Corrections:</p> <ul style="list-style-type: none"> Define Partial Header Encryption Mode field for ↓↓Like↓ <ins>↑↑Link↑</ins> IDE Correct Partial Header Encryption Algorithm to correct functional issues and to use Byte level granularity instead of bit granularity as required in AES-GCM. Update <u>Figure L-1</u> to include Attr[2] (a.k.a. A2 or IDO). Add <u>Section L.2</u>. |
| 6.0.1 | A27 | Clarify Flit Mode Receiver parsing requirements regarding of the Length field. |
| 6.0.1 | A35 | <p>Shared Flow Control Corrections:</p> <ul style="list-style-type: none"> Split Table 3-2 into <u>Table 3-2</u> and <u>Table 3-3</u> Correct encoding of Symbol +0 bit 3 in <u>Figure 3-9</u>, <u>Figure 3-10</u>, and <u>Figure 3-11</u> to match <u>Table 3-3</u> Remove obsolete “Shared Flow Control Flit_Marker” text from the Recommended Priority of Scheduled Transmissions Implementation Note. |
| 6.0.1 | A35a | Shared Flow Control Usage Limit is <u>RsvdP</u> for Non-Flit Mode components. |
| 6.0.1 | A37 | Corrections to the Flit Sequence Number and Retry Mechanism (see <u>Section 4.2.3.4.2.1</u>) |
| 6.0.1 | A38 | <p>L0p Corrections:</p> <ul style="list-style-type: none"> Preserve LFSR relationship between Lanes on width increase Update scrambler behavior during L0p width increase Update Link ↓↓Management↓ <ins>↑↑Management↑</ins> DLLP Behavior Update LFSR advancement rules for L0p |
| 6.0.1 | A53 | Define the <u>Flit Error Counter Interrupt Enable</u> bit in the <u>Flit Error Counter Control Register</u> . |

Base 6.4 vs Base 6.3

| Revision | Errata | Description |
|----------|--------|--|
| 6.0.1 | A62 | Clarify OHC rules. |
| 6.0.1 | A67 | Training Set Corrections: <ul style="list-style-type: none">• Clarify equalization fields in TS0 Ordered Sets (see § Table 4-40). |
| 6.0.1 | A73 | Modified TS1/TS2 corrections |
| 6.1 | A392 | Clarify End-to-End TLP Prefix Support indications to software. |
| 6.1 | A393 | Corrections to Flow Control rules <ul style="list-style-type: none">• To clarify that when [Merged] is used, distinct ↑↑UpdateFCs↓ ↑↑UpdateFC_DLLPs↑ are sent for Posted and Completion credits. This provides buffer consumption visibility to the transmitter allowing a vendor-specific Quality of Service (QoS) mechanism to be implemented.• Clarify definition of FC Unit Size• Require consistency across VCs for shared credits – for example, if one VC advertises [Infinite.3], all VCs must also advertise it• Update minimum flow control credit rules.• Clarify scale factor rules when [Zero] or [Merged] are used.• Update § Equation 2-8• Add § Equation 2-9• Update § Equation 2-10• Add § Equation 2-11• Update § Equation 2-16• Add § Equation 2-17 and § Equation 2-18 |
| 6.2 | A531 | Correct ↑↑§ Figure 6-64↑ “IDE TLP Prefix (NFM)” and clarify text. |
| 6.2 | A540 | Corrections to Equalization Phase 1 rules. |
| 6.2 | A544 | Corrections to Flit Ack and Nak rules. |
| 6.2 | A577 | Define EIE pattern for Retimers for 1b/1b encoding. |
| 6.2 | A578 | Required modifications for Retimer timeouts for 64.0 GT/s. |
| 6.2 | A587 | Correct inconsistencies between decimal and binary values listed in § Table 2-5 “Flit Mode TLP Header Type Encodings”. |
| 6.3 | A656 | Sending NOP Flits with explicit sequence number set to NEXT_TX_FLIT_SEQ_NUM - 1 during Replay could cause issues at the far end receiver, especially if next Flit has an implicit sequence number. |
| 6.3 | A687 | IMPLICIT_RX_FLIT_SEQ_NUM Rules must ignore invalid Flits. |

Base 6.4 vs Base 6.3

Important Errata and Approved ECNs

| Revision | Errata / ECN | Description |
|----------|--------------|---|
| 6.0.1 | B5 | Clarify value of the Alternate Protocol Count field in the Alternate Protocol Capabilities Register. |
| 6.0.1 | B7 | Incorrect capability name used in the description of the Capability ID field of Lane Margining at the Receiver Extended Capability Header. |
| 6.0.1 | B8 | Clarify that behavior is undefined when Message Data Register for MSI low bits conflict with the MME setting. |
| 6.0.1 | B10 | A single PCI Express Function is permitted to contain multiple VSEC structures. |
| 6.0.1 | B11 | Clarify Retimer Loopback behavior. |
| 6.0.1 | B12 | Recommend that parity errors in Control SKP OS be checked only when immediately preceded by a Data Block. |
| 6.0.1 | B13 | Clarify Alternate Protocol Negotiation Status error values. |
| 6.0.1 | B14 | Define what it means for Scaled Flow Control to be “activated”. |
| 6.0.1 | B15 | Margining Ready applies to speeds above 16.0 GT/s. |
| 6.0.1 | B16 | Update Link Number encoding in § Table 4-39. |
| 6.0.1 | B17 | Add Implementation Note: Implementation Note: Delays in Data Link Layer Link Active Reflecting Link Control Operations |
| 6.0.1 | B18 | Clarify Receiver Impedance Propagation Rules. |
| 6.0.1 | B21 | Update recommended behavior on transitions from Hot Reset to Detect.Active. |
| 6.0.1 | B22 | Clarify Flit Mode poison rules. |
| 6.0.1 | B23 | <ul style="list-style-type: none"> Shrink figures § Figure 3-9 and § Figure 3-10 so they're not cut off. Redraw § Figure 4-27 so it's readable. |

Figure 1 Old Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Transmit side §

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|-------------|
| | | |

Figure 2 New Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Transmit side §

- Redraw § Figure 4-28 so it's readable.

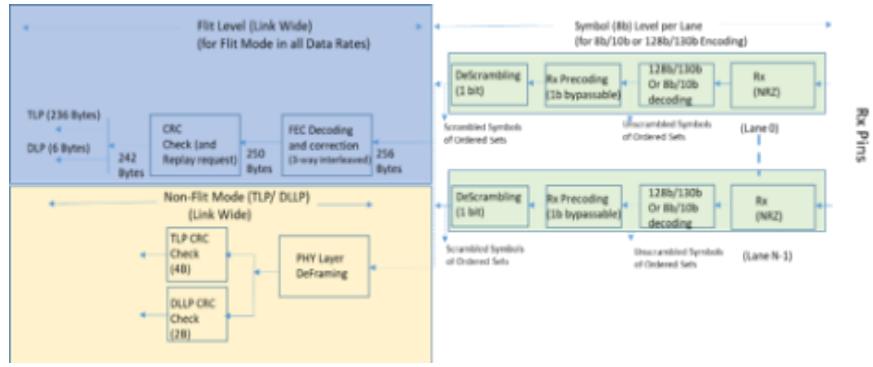


Figure 3 Old Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Receive side §

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| | | |
| | | <p style="text-align: center;"><i>Figure 4 New Figure: Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Receive side §</i></p> |
| 6.0.1 | B24 | Lane Margining at the Receiver Corrections |
| 6.0.1 | B25 | Define <u>FLIT</u> in Terms and Acronyms |
| 6.0.1 | B26 | Completion Timeout Ranges Supported , Completion Timeout Disable Supported , and Completion Timeout Value only apply to Non-Posted Requests |
| 6.0.1 | B28 | Requester must accept any Segment number in completions. |
| 6.0.1 | B29 | TC field must be 000b in § Figure 2-72 , § Figure 2-74 , § Figure 2-76 , and § Figure 2-78 . Type field was incorrect in § Figure 2-74 and § Figure 2-78 . |
| 6.0.1 | B30 | Routing field was incorrect in § Table 2-36 IDE Messages. |
| 6.0.1 | B31 | Clarify origins of T_{rst} value of 1 ms. |
| 6.0.1 | B34 | § Table 4-36 TS1 Ordered Set in 8b/10b and 128b/130b Encoding: Symbol 4, Bit 7 had the wrong LTSSM state for the speed_change usage. |
| 6.0.1 | B40 | Update the exponents of α in § Figure 4-56 . α^{85} and α^{84} changed to α^{84} and α^{83} respectively. Update exponents of α in Column B 0 in § Figure 4-57 . α^{85} through α^{92} changed to α^{84} through α^{91} respectively. |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|-------------|
| | | |

| | B ₀ | ... | B ₈₂ | B ₈₃ | B ₈₄ | B ₈₅ |
|---|----------------|-----|-----------------|-----------------|-----------------|-----------------|
| 0 | α^{85} | ... | α^2 | α | 1 | p_0 |
| 1 | α^{86} | ... | α^3 | α^2 | α | p_1 |
| 2 | α^{87} | ... | α^4 | α^3 | α^2 | p_2 |
| 3 | α^{88} | ... | α^5 | α^4 | α^3 | p_3 |
| 4 | α^{89} | ... | α^6 | α^5 | α^4 | p_4 |
| 5 | α^{90} | ... | α^7 | α^6 | α^5 | p_5 |
| 6 | α^{91} | ... | α^8 | α^7 | α^6 | p_6 |
| 7 | α^{92} | ... | α^9 | α^8 | α^7 | p_7 |

Check Bits

Row Parity [N:3:0]

Figure 5 Old Figure: Powers of alpha for the check bits for Bytes 0 to 84 §

| | B ₀ | ... | B ₈₂ | B ₈₃ | B ₈₄ | B ₈₅ |
|---|----------------|-----|-----------------|-----------------|-----------------|-----------------|
| 0 | α^{84} | ... | α^2 | α | 1 | p_0 |
| 1 | α^{85} | ... | α^3 | α^2 | α | p_1 |
| 2 | α^{86} | ... | α^4 | α^3 | α^2 | p_2 |
| 3 | α^{87} | ... | α^5 | α^4 | α^3 | p_3 |
| 4 | α^{88} | ... | α^6 | α^5 | α^4 | p_4 |
| 5 | α^{89} | ... | α^7 | α^6 | α^5 | p_5 |
| 6 | α^{90} | ... | α^8 | α^7 | α^6 | p_6 |
| 7 | α^{91} | ... | α^9 | α^8 | α^7 | p_7 |

Check Bits

Row Parity [N:3:0]

Figure 6 New Figure: Powers of alpha for the check bits for Bytes 0 to 84 §

| | | |
|-------|-----|--|
| 6.0.1 | B44 | Extended Synch is not used for L0s in Flit Mode, but is still relevant for other purposes. |
| 6.0.1 | B45 | Add 64.0 GT/s encoding to Enable SKP OS Generation Vector . |
| 6.0.1 | B48 | Clarify 1b/1b encoding EIOSQ terminology in Compliance and Modified Compliance Patterns. |
| 6.0.1 | B49 | Execute Requested is reserved in all Memory Requests other than Untranslated Memory Read Requests. |
| 6.0.1 | B50 | Typo in units column of § Table 8-16 , Parameter T _{TX-CH-UPW-RJ-64G} |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|---|
| 6.0.1 | B51 | <ul style="list-style-type: none"> • Typo in units column of § <u>Table 8-18</u>, Parameter <u>Z_{C-DC}</u> • Locate artwork for § <u>Figure 8-86</u> |
| 6.0.1 | B54 | Don't send Selective IDE Stream IDE Sync and IDE Fail messages unless the V bit is Set for that RID Base . |
| 6.0.1 | B55 | The "subsequent IDE TLPs" error reporting language applies regardless of how Insecure was reached. |
| 6.0.1 | B56 | Selective IDE Stream Enable and Link IDE Stream Enable are edge triggered. |
| 6.0.1 | B58 | The Selective IDE Stream Control Register , Default Stream field only applied to Selective IDE Streams. |
| 6.0.1 | B59 | Routed to Root Complex is permitted only when the Partner Port is a Root Port, as indicated by the Default Stream bit being Set. |
| 6.0.1 | B60 | Clarify rules for selecting the Selective IDE Stream for a given TLP. |
| 6.0.1 | B64 | Clarify Segment number processing rules in Flit Mode Completer Transaction ID processing. |
| 6.0.1 | B66 | <p>The <i>Data Link Feature Exchange Enable</i> field was renamed to Data Link Feature Exchange Is Enabled .</p> <p>This errata also applies to [PCIE-5.0].</p> |
| 6.0.1 | B68 | Correct Type field value in § <u>Figure 10-13</u> (was 0000 0000b should be 0000 0011b). |
| 6.0.1 | B69 | In Shared Flow Control Usage Limit Enable , clarify wording to indicate which Shared Flow Control Usage Limit is affected (duplicate field names in VC Resource Control Register and MFVC VC Resource Control Register). |
| 6.0.1 | B70 | DC Balance correction in 1b/1b encoding of TS0 , TS1 , and TS2 ↑↑↑ |
| 6.0.1 | B71 | Consolidate description of 1b/1b behavior for SKP Ordered Sets ↑↑↑ |
| 6.0.1 | B72 | TS2 Symbol 6, operating at 2.5 or 5.0 GT/s and Requesting equalization ↑↑32.0 GT/s↓ ↑↑32.0 GT/s↑ encoding is only for 32.0 GT/s (was "32.0 GT/s or higher ") |
| 6.0.1 | B75 | Clarify behavior in CMA-SPDM Rules. |
| 6.0.1 | B77 | Update artwork for § <u>Figure 6-79</u> . |
| 6.0.1 | B79 | In Flit Mode, the 16.0 GT/s Local Data Parity Mismatch Status Register , 16.0 GT/s First Retimer Data Parity Mismatch Status Register , and 16.0 GT/s Second Retimer Data Parity Mismatch Status Register are used at all 128b/130b and 1b/1b data rates. |
| 6.0.1 | B81 | Clarify 128b/130b Control and Standard SKP Alternation Rule interactions with L0p |
| 6.0.1 | B82 | Editorial: SRIS_Mode_Enabled is a variable, not a bit. |
| 6.0.1 | B83 | Define CXL Src bit in ATS Translation Requests. This is bit 3 in byte 15 in § <u>Figure 10-10</u> , § <u>Figure 10-11</u> , § <u>Figure 10-12</u> , and § <u>Figure 10-13</u> . Editorial change in § <u>Table 10-3</u> . |
| 6.0.1 | B85 | Loopback Follower changes |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| 6.0.1 | B86 | <ul style="list-style-type: none"> • § Table 4-39 Equalization Byte 0 correction (Symbols 1,9) • § Table 4-37 Symbol 6 8.0 GT/s should be 8.0 GT/s or higher • § Section 4.2.7.4.4 , Width change rule change, change “128b/138b” to “128b/130b or 1b/1b” (3 places) |
| 6.0.1 | B87 | Update Recovery.RcvrLock behavior when hardware autonomous equalization is not adopted |
| 6.0.1 | B88 | Define 1b/1b Phy Payload type Alternation Rule |
| 6.0.1 | B89 | <ul style="list-style-type: none"> • Change name of L0p Supported capability bit (was Receiver L0p Supported) • Clarify Target Link Width behavior • Clarify L0p.Priority behavior |
| 6.0.1 | B90 | Clarify Receiver Framing Requirements (including L0p in Flit Mode) |
| 6.0.1 | B92 | Clarify that the LTSSM must not advertise speeds above 32.0 GT/s in Polling.Active and Polling.Configuration . During those states, negotiation for greater than 32.0 GT/s is ↑↑occurring↓ <ins>↑↑occurring↑</ins> |
| 6.0.1 | B93 | Add implementation note “Reject Coefficient Values with TS0 Ordered Sets” in § Section 4.2.7.4.2.2.1 . |
| 6.0.1 | B94 | Attr[2] was incorrectly shown as Reserved in § Figure 10-6 , § Figure 10-7 , § Figure 10-10 , and § Figure 10-11 . |
| 6.0.1 | B95 | Clarify historical ↑↑timestamp↓ <ins>↑↑timestamp↑</ins> invalidation behavior in § Section 6.21.3.2 . |
| 6.0.1 | B96 | Correct the register width of the Root Capabilities Register , § Table 7-31 . Was 32 bits, but the register is actually 16 bits wide. |
| 6.0.1 | B98 | Update § Figure 7-229 and § Table 7-208 to better reflect that the Virtual Channel Extended Capability uses Capability ID 0002h or 0009h. |
| 6.0.1 | B99 | Clarify that intermediate receivers should not check crossing 4 KB boundaries. Implementing this check could invalidate such silicon changes for future TLP Type definitions (e.g., such future TLP Types could be similar to the existing Atomic CAS where the relationship between the TLP Length field and the affected memory range is not 1:1). |
| 6.0.1 | B100 | Clarify that L0p is supported at all data rates. |
| 6.0.1 | B102 | Segment was erroneously left out of IDE RID Association mechanism |
| 6.0.1 | B103 | Clarify IDE TLP Aggregation rules. See § Section 6.33.6 . |
| 6.0.1 | B104 | Add implementation note SKP Ordered Sets in a Data Stream in Flit Mode to § Section 4.2.8.5 . |
| 6.0.1 | B106 | Clarify the FM/NFM rules for PCRC. |
| 6.0.1 | B107 | Clarify that IDE TLPs in FM always have OHC-C. |
| 6.0.1 | B108 | CMA-SPDM state should not be affected by FLR. |
| 6.0.1 | B111 | Clarify Retimer response for Access Retimer Register command. |
| 6.0.1 | B112 | Add implementation note Security Issues Associated with Non-Enabled Bytes . |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| | | The following figures incorrectly described reserved bits as RsvdP that should be RsvdZ . some of these were also incorrect in [PCIE-5.0] but were correct in [PCIE-4.0]. <ul style="list-style-type: none"> • § Figure 7-32 – Slot Status Register • § Figure 7-124 – Device Status 3 Register • § Figure 7-382 – Link IDE Stream Status Register • § Figure 7-385 – Selective IDE Stream Status Register |
| 6.0.1 | B113 | |
| 6.0.1 | B114 | Add new § Section 4.2.18.3 describing Lane Margining contents for Flit Mode Control SKP Ordered Sets at 8.0 GT/s. |
| 6.0.1 | B115 | Clarify the Variant bit rules in § Section 2.7.1 . |
| 6.0.1 | B116 | OHC-A1 is also used for Route to Root Complex Messages (e.g., Page Request Messages) and for Translation Requests that Set NW |
| 6.0.1 | B117 | Clarify that "overflow" for the PR Received Counters (NPR/CPL) means the 64b counter itself overflows. |
| 6.0.1 | B118 | Clarify that, with system support for DRS, Devices are permitted to take longer than 1 second to become Configuration-Ready. |
| 6.0.1 | B119 | Clarify text in Segment Captured bit description. |
| 6.0.1 | B120 | When the length of an ATS Request is improper, the result should be undefined, not a Malformed TLP. |
| 6.0.1 | B121 | Clarify TLP Prefix Processing rules for Flit Mode. |
| 6.0.1 | B122 | Clarify completion rules for TLPs with Reserved Type field values. |
| 6.0.1 | B123 | Deprecate Flit Mode TLP Suffixes. They were unused and have latency concerns. |
| 6.0.1 | B124 | Clarify that Equalization Bypass to Highest NRZ Rate Disable is optional. |
| 6.0.1 | B127 | Clarify L0p Framing Error rule. Clarify L0p ↓↑retimer↓ <ins>↓↑Retimer↓</ins> behavior. |
| 6.0.1 | B128 | Update the Compliance Pattern for 1b/1b operation. |
| 6.0.1 | B130 | Clarify Loopback.Active language. |
| 6.0.1 | B131 | <ul style="list-style-type: none"> • Deprecate Report Longest Burst vs First Burst • Flit Error Counter does not roll over • Introductory text in § Section 7.7.8.7 describes all FBER Measurement Status Registers (i.e., FBER Measurement Status 1 Register through FBER Measurement Status 10 Register) • Remove Pseudo Port wording in Lane #0 Correctable Counter (also applies to the rest of the Lane counters). |
| 6.0.1 | B132 | <p>Correct Tag field usage in Messages and Vendor-Defined Messages.</p> <ul style="list-style-type: none"> • Message byte 6 is Reserved unless specifically defined. |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| | | <ul style="list-style-type: none"> Reserved bits must be copied intact during translation to/from Flit Mode and Non-Flit Mode. Change wording for byte 6 of Non-Flit Mode Messages § Figure 2-49. Change wording for byte 6 of Flit Mode Messages § Figure 2-50. Low 7 bits of byte 6 are available for use in Vendor-Defined Messages. Change wording for byte 6 of Non-Flit Mode Vendor-Defined Messages § Figure 2-53 Change wording for byte 6 of Flit Mode Vendor-Defined Messages § Figure 2-54 <p>This errata also applies to [PCIE-5.0].</p> |
| 6.1 | B305 | <p>Clarify flit error counting Behavior</p> <ul style="list-style-type: none"> Events to Count field of the Flit Error Counter Control Register § Section 4.2.3.4.2 § Section 4.2.5.1 |
| 6.1 | B306 | Electrical Idle exit timing changes |
| 6.1 | B308 | Clarify implementation note Detection of Improper Reordering in IDE TLP Sub-Streams (§ Section 6.33.5) |
| 6.1 | B309 | Support 64.0 GT/s in Loopback.Entry and Detect.Quiet |
| 6.1 | B310 | Clarify Recovery.RcvrLock , Recovery.RcvrCfg , Recovery.Speed , and Recovery.Idle rules |
| 6.1 | B311 | Text was misplaced in Recovery.RcvrCfg |
| 6.1 | B312 | L0p rules |
| 6.1 | B314 | Lane Numbering matching rules in 1b/1b |
| 6.1 | B315 | Centralize text describing Default Lane numbers |
| 6.1 | B316 | IDE Terminus rules for Switch / Root Port |
| 6.1 | B317 | Add implementation note Determination of Slot Number Information |
| 6.1 | B319 | <ul style="list-style-type: none"> Update Symbol Time definition for 1b/1b encoding Update Retimer Latency Limit values for SRIS § Table 4-79 |
| 6.1 | B320 | Provide Test Keys for testing IDE in § Section L.5 |
| 6.1 | B322 | Recovery.RcvrLock to Recovery.Equalization transition |
| 6.1 | B323 | SRIS Clocking bit is also meaningful in Flit Mode with 8b/10b and 128b/130b encoding |
| 6.1 | B325 | Recovery.Idle to Loopback |
| 6.1 | B326 | Configuration.Linkwidth.Start rules for 64.0 GT/s |
| 6.1 | B328 | Clarify Requester Segment field behavior in OHC-C |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| 6.1 | B329a | Configuration Requests cannot map to the Default IDE Stream |
| 6.1 | B330 | Clarify definition of Flit halves |
| | | Corrections to EP and Length fields of the following messages: <ul style="list-style-type: none">• § Figure 10-22• § Figure 10-24• § Figure 10-26• § Figure 10-28• § Figure 2-72• § Figure 2-74• § Figure 2-76• § Figure 2-78 |
| 6.1 | B331 | |
| 6.1 | B332 | Function Level Reset clarification |
| 6.1 | B335 | Recovery.RcvrLock transmitter equalization coefficients |
| 6.1 | B336 | EIEOS inference behavior only occurs when the LTSSM rules already allow entry to Recovery |
| 6.1 | B338 | Define terms Endpoint Upstream Port and Switch Port . |
| 6.1 | B340 | Clarify rules for where a PASID Extended Capability is permitted and define associated behavior. |
| 6.1 | B341 | Errata against TDISP ECN-Link-Activation-07-Dec-2017 Poisoned TLP rules in TDISP |
| 6.1 | B342 | Returning TC0/VC0 to secure state in Link IDE |
| 6.1 | B344 | TEE rules when not using TDISP |
| 6.1 | B345 | Remove confusing and ↑↓unnecessary↓ <ins>↑↑unnecessary↑</ins> text regarding CMA-SPDM |
| 6.1 | B346 | Remove confusing and ↑↓unnecessary↓ <ins>↑↑unnecessary↑</ins> text regarding CMA-SPDM |
| 6.1 | B347 | Update § Figure 6-63 to recommend setting IDE Enable last when programming keys. Add text to strongly recommend this behavior while permitting otherwise. |
| 6.1 | B348 | Errata against TDISP ECN DOE errors don't usually impact TDISP |
| 6.1 | B349 | ERRATA against TDISP ECN Add examples applicable IDE errors. |
| 6.1 | B353 | Translation Requests always contain OHC-A1 as they always contain the NW bit. |
| 6.1 | B354 | Correct naming of the ↑↓Received IDE Fail Message↓ <ins>↑↑Received IDE Fail Message↑</ins> in Link IDE Stream Status Register and Selective IDE Stream Status Register |
| 6.1 | B355 | Clarify requirements on system software behavior ↑↓during↓ <ins>↑↑during↑</ins> IDE enablement. |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|---|
| 6.1 | B356 | Update precoding behavior |
| 6.1 | B357 | Clarify Alternate Protocol Negotiation behavior |
| 6.1 | B358 | Add ↓↑Implementation note↓↑Handling TLPs Spanning Multiple Flits↓↑ <ins>↑↑Implementation Note: Handling TLPs Spanning Multiple Flits↑↑</ins> |
| 6.1 | B359 | Clarify wording for Gray coding of DC balance symbol |
| 6.1 | B360 | Clarify K_SET_GO behavior |
| 6.1 | B361 | Errata against TDISP ECN Reword <u>§ Section 11.2.1</u> to reflect T-Bit behavior required by TDISP |
| 6.1 | B363 | Flit Mode Status has different behavior for Upstream and Downstream Ports |
| 6.1 | B365 | Clarify precoding text for 64.0 GT/s |
| 6.1 | B367 | Clarify All VCs Enabled wording in the <u>Port VC Control Register</u> and <u>MFVC Port VC Control Register</u> |
| 6.1 | B369 | Add rules in <u>§ Section 6.33.8</u> |
| 6.1 | B370 | Clarify Latency Tolerance ↓↑Reporting↓↑Reporting↑ behavior when entering and exiting D0 |
| 6.1 | B373 | Clarify Retimer rules for Hot Reset |
| 6.1 | B374 | Clarify rules for <u>Loopback.Entry</u> |
| 6.1 | B375 | Clarify rules for <u>Loopback.Entry</u> |
| 6.1 | B376 | Clarify wording in Phase 0 and Phase 1 of Transmitter Equalization |
| 6.1 | B377 | Clarify rules for <u>Loopback.Entry</u> |
| 6.1 | B378 | Clarify Ordered Set Error Injection behavior |
| 6.1 | B381 | Clarify Local TLP Prefix rules in Flit Mode TLP Grammar |
| 6.1 | B382 | Correct uniqueness rules for Transaction IDs |
| 6.1 | B384 | <u>Recovery.Speed</u> corrections |
| 6.1 | B385 | Correct Flit Ack, Nak, and Discard rules |
| 6.1 | B386 | Add 64.0 GT/s to <u>Recovery.RcvrLock</u> rule |
| 6.1 | B388 | Framing Error in the context of <u>§ Section 4.2.5.8</u> is only for Non-Flit Mode |
| 6.1 | B394 | Update Configuration Request Routing Rules for IDE |
| 6.1 | B395 | K_SET_GO or K_SET_STOP behavior with an invalid ↓↑Stream ID↓↑Stream_ID↑ <ins>↑↑Stream_ID↑↑</ins> |
| 6.1 | B397 | L0p ↓↑beahvior↓↑behavior↑ when an <u>EIEOSQ</u> is scheduled near an <u>SDS Ordered Set</u> |
| 6.1 | B398 | Flit Sequence Number and Retry Mechanism |
| 6.1 | B405 | <u>VF Enable</u> behavior |
| 6.1 | B406 | Clarify rules for Local TLP ↓↑PRefixes↓↑Prefixes↑ <ins>↑↑Prefixes↑↑</ins> and ↓↑Lpcal↓↑Local↑ <ins>↑↑Local↑↑</ins> TLPs |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| 6.1 | B407 | Clarify L0 behavior when receiving EIOS behavior L0 interaction <ins>with L0p</ins> |
| 6.1 | B408 | Flit Sequence Number rules |
| 6.1 | B409 | In § Section 4.2.7.4.4, the LTSSM must not initiate a width change if speed_change is set to 1b |
| 6.1 | B412 | Errata to TDISP ECN FLR to Functions other than Function 0 do not affect SPDM or IDE streams. |
| 6.1 | B414 | Errata to UIO ECN TC to VC Mapping rules |
| 6.1 | B415 | Behavior for L0s in Flit Mode |
| 6.1 | B416 | Clarify wording around Hot and Warm Reset |
| 6.1 | B418 | Recommend software recovery procedure for use when the page request queue overflows This Errata is part 1 of 10 that constitute ATS 1.2 |
| 6.1 | B419 | U-bit will eventually be deprecated This Errata is part 2 of 10 that constitute ATS 1.2 |
| 6.1 | B420 | Address space overlap clarification This Errata is part 3 of 10 that constitute ATS 1.2 |
| 6.1 | B421 | ITag uniqueness rules This Errata is part 4 of 10 that constitute ATS 1.2 |
| 6.1 | B422 | ATS and PRI interactions with Shadow Functions This Errata is part 5 of 10 that constitute ATS 1.2 |
| 6.1 | B423 | ATS Translation Completion Status updates This Errata is part 6 of 10 that constitute ATS 1.2 |
| 6.1 | B425 | Invalidate Queue Depth , Page Aligned Request , and Global Invalidate Supported updates This Errata is part 7 of 10 that constitute ATS 1.2 |
| 6.1 | B426 | Execute Requested update This Errata is part 8 of 10 that constitute ATS 1.2 |
| 6.1 | B427 | Execute Permission Supported update This Errata is part 9 of 10 that constitute ATS 1.2 |
| 6.1 | B428 | ATS Enable update This Errata is part 10 of 10 that constitute ATS 1.2 |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|---|
| 6.1 | B429 | Scaled Flow Control updates |
| 6.1 | B430 | Data Link Feature is required in Flit Mode |
| 6.1 | B431 | Errata against UIO ECN: Remove the sentence “Because UIO can only be enabled when end-to-end UIO support exists, it is never necessary for the Root Complex to take ownership of UIO Requests.” from the Implementation Note Root Complex Support for peer-to-peer Non-Posted Memory Transactions that Traverse Hierarchies. |
| 6.1 | B432 | Mandate capacitor placement on the Transmit end of a Link. |
| 6.1 | B433 | Clarify valid tag values for 14-bit tags. |
| 6.1 | B434 | Clarify rules for Switch FM to NFM translation. |
| 6.1 | B435 | Address Routed Message corrections in Flit Mode. |
| 6.1 | B436 | Clarification of Tag keep-out ranges. |
| 6.2 | B500 | Clarify rules for PASID application. |
| 6.2 | B501 | Clarify rules for Address Routed Messages. |
| 6.2 | B502 | Clarify OHC-A5 requirements for Selective IDE Completions. |
| 6.2 | B503 | Remove needless indirection regarding requirement that VFs must not implement PASID Extended Capability. |
| 6.2 | B506 | Clarify rules for TDI regarding MSI/MSI-X. |
| 6.2 | B507 | Errata B507: Remove Implementation Note “FUTURE TEE EXTENSIONS”: TEE extensions for sub-function I/O-virtualization techniques will be covered in a future revision of this specification. |
| 6.2 | B508 | Remove unnecessary restriction on VC1 in SVC. |
| 6.2 | B510 | Correct typo in Recovery.RcvrCfg. |
| 6.2 | B512 | Clarify application of IDE Default Stream to Translation Requests and Untranslated Memory Requests. |
| 6.2 | B513 | Clarify application of ordering rules for Flow-Through Selective IDE. |
| 6.2 | B515 | Clarify Flit Performance Measurement mechanisms. |
| 6.2 | B515a | Clarify Segment rules for Flit Mode Requesters/Completers within an RC. |
| 6.2 | B517 | Correct § Figure 6-46 “CMA-SPDM as Part of a Layered Architecture”. |
| 6.2 | B518 | Correct use of “Precision Time Measurement”. |
| 6.2 | B520 | Clarify Phase 1 of Transmitter Equalization text. |
| 6.2 | B522 | Correct paragraph alignment. |
| 6.2 | B524 | Clarify rules for EQ TS2 Ordered Set use. |
| 6.2 | B525 | Add LOp register cross-references. |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| 6.2 | B529 | Clarify earmarked TLP handling. |
| 6.2 | B535 | FLIT_REPLAY_NUM clarifications. |
| 6.2 | B536 | Clarify Segment Base programming for certain scenarios. |
| 6.2 | B537 | Clarify rules for Selective IDE for Configuration Requests. |
| 6.2 | B538 | Clarify Privileged Mode Requested and Execute Requested/Execute Permitted rules corrupted during Revision 6.0 development. |
| 6.2 | B541 | Clarify L0p request handling rules. |
| 6.2 | B546 | <p>Clarify figures and text regarding TLP Header Byte 6 for Message Requests.</p> <p>The following NFM Message Request Header figures in § Section 2.2.8 have byte 6 labeled as “Tag”. Change the label to “R” or “Reserved” (based on what fits). Note: Figures are automatically generated and these changes are not marked in the document.</p> <ul style="list-style-type: none"> • § Figure 2-51 • ↑↑§ Figure 2-55↑ • ↑↑§ Figure 2-63↑ • ↑↑§ Figure 2-65↑ • ↑↑§ Figure 2-67↑ • ↑↑§ Figure 2-68↑ |
| 6.2 | B547 | Clarify AT field requirements for Messages Routed by Address. |
| 6.2 | B548 | Clarify MMB Capabilities Register language. |
| 6.2 | B549 | Clarify handling of received DMWr Requests. |
| 6.2 | B550 | Clarify and cross-reference existing rules regarding Segment exceptions for Root Ports. |
| 6.2 | B551 | Clarify SKP Ordered Set rules with L0p . |
| 6.2 | B552 | Correct omission of TS0 in rules for Phase 1 of Transmitter Equalization. |
| 6.2 | B553 | Clarify text regarding Electrical Idle Sequences on entry to Recovery.RcvrLock . |
| 6.2 | B554 | Clarify Autonomous Change / Selectable De-emphasis for TS1 / TS2 Ordered Set with 1b/1b Encoding. |
| 6.2 | B555 | Clarify content of TS2 Ordered Set with 1b/1b Encoding. |
| 6.2 | B556 | Clarify handling of NOP2 DLLP during equalization. |
| 6.2 | B557 | Clarify Flit Replay Transmit rules. |
| 6.2 | B559 | UIO clarifications. |
| 6.2 | B561 | Clarify and cross-reference DMWr capability reporting. |
| 6.2 | B562 | OHC-E clarifications. |
| 6.2 | B564 | Clarify L0p Link width rules. |
| 6.2 | B565 | Clarify rules for handling an invalid Flit Sequence Number in a received Ack Flit or Nak Flit. |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| 6.2 | B566 | Clarify Retimer data rate change and determination rules. |
| 6.2 | B568 | Correct Bad DLLP entry in § Table 6-4 “Data Link Layer Error List”. |
| 6.2 | B569 | Clarify handling of ID Routed Messages targeting a Function that is not implemented. |
| 6.2 | B571 | <p>Correct figures to reduce the Key Sub-Stream field width to 3 bits. This affects:</p> <ul style="list-style-type: none"> • § Figure 6-57 • § Figure 6-58 • § Figure 6-59 • § Figure 6-60 • § Figure 6-61 |
| 6.2 | B572 | Clarify rules for implementing <u>PTM Extended Capability</u> . |
| 6.2 | B573 | Clarify Data Stream, and rules for choosing Flit Mode vs. Non-Flit Mode. |
| 6.2 | B575 | Clarify <u>Half Scrambling</u> |
| 6.2 | B576 | Add Implementation Note: Forwarding D21.3 Symbol with Incorrect Disparity. |
| 6.2 | B580 | Clarify rules to detect an <u>SDS</u> . |
| 6.2 | B581 | Add Implementation Note <u>Orthogonality of L0p and L1/L2</u> |
| 6.2 | B583 | Clarify definition of UIO TLPs. |
| 6.2 | B583a | Clarify that Translation Requests are not defined for UIO Memory Reads. |
| 6.2 | B585 | Correct section references. |
| 6.2 | B586 | Clarify definition of Idle Flit, <u>Nop Flit</u> , and Payload Flit. |
| 6.2 | B588 | DPC doesn't invent Completions for UIO Requests. |
| 6.2 | B595 | Clarify Larger Tags rules. |
| 6.2 | B599 | Clarify Precoding negotiation. |
| 6.3 | B600 | Effect of <u>Detect</u> on Precoding |
| 6.3 | B601 | Population of TS1/TS2 Ordered Set fields during L0p |
| 6.3 | B602 | Retimer changes to Enter Compliance Rules |
| 6.3 | B603 | Correct Flit Layout figures to use Symbol Times instead of UI. With PAM4, UI was no longer accurate. |
| 6.3 | B604 | Add Transmitter Preset rules for Phase 0 of Transmitter Equalization when sending TS0 Ordered Sets |
| 6.3 | B605 | <p>Change label for rightmost block in § Figure 4-22 from “PAM4 voltage Conversion to 2-bit aligned” to “2-bit aligned to PAM-4 voltage Conversion”</p> <p>Update fonts in § Figure 4-22 and § Figure 4-23</p> |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|---|
| 6.3 | B606 | Clarify TS0 rules for Phase 0 of Transmitter Equalization TS0s don't have the <u>Reject Coefficient Values bit</u> , so references to setting it need to be qualified by TS1 Ordered Sets, and the text needs to say what happens when transmitting TS0s. |
| 6.3 | B607 | Clarify rules for determining behavior for coalesced UIO Completions with mixed Completions Status values |
| 6.3 | B608 | Correction of Electrical Idle time rules in <u>Polling.Compliance</u> |
| 6.3 | B616 | Remove leftover text referring to Receiver Preset Hints (that are only present at 8.0 GT/s). Update text to use correct field names. |
| 6.3 | B617 | Clarify SKP OS behavior in Rules for Transmitters |
| 6.3 | B618 | Define Flit Mode behavior when TPH is used and Byte Enable values do not match the Non-Flit Mode implied values (see § <u>Section 2.2.5.1</u>). |
| 6.3 | B619 | Clarify Alternate Protocol Negotiation when more than one alternate protocol is supported (i.e., PCIe plus two or more alternate protocols). |
| 6.3 | B621 | Defined minimum Receiver rules for detecting Idle Flits. |
| 6.3 | B623 | Clarify behavior when changing <u>VF Enable</u> . <u>Immediate Readiness</u> does not apply to VFs. |
| 6.3 | B624 | Define the terms <u>RefClk</u> , <u>recommended</u> , and <u>strongly recommended</u> |
| 6.3 | B625 | Clarify software rules for interpreting Flit Error Log registers. |
| 6.3 | B626 | Add implementation note <u>Summary of L0p Transmitter/Receiver Behavior</u> . |
| 6.3 | B626a | Clarify <u>L0p Enable</u> rules. Note: This Erratum adds two sentences to the implementation note <u>§ Implementation Note: Summary of L0p Transmitter/Receiver Behavior</u> located immediately before <u>§ Table 4-55</u> . That change may be marked as part of Erratum B626. |
| 6.3 | B628 | Cleanup terminology regarding Update Flow Control and Optimized_Update_FC |
| 6.3 | B630 | Remove values for fields Fmt and Type in <u>§ Figure 10-6</u> and <u>§ Figure 10-7</u> . MRd and MWr are not the only types of Memory Requests. |
| 6.3 | B632 | Flit Latency Tracking Counter unit changed to ns from µs. Incomplete integration error of earlier Erratum B515. This unfortunately creates a situation where a software workaround is needed for <u>Flit Latency Tracking Counter</u> use. |
| 6.3 | B633 | Clarify rules for when N_FTS is meaningful in <u>TS1</u> , <u>TS2</u> , and <u>Modified TS1/TS2</u> Ordered Sets. |
| 6.3 | B634 | Relax definition of L0p Exit Latency encodings. Indicate that L0p Exit Latency is a hint (e.g., there is no compliance issue if latency is longer). |
| 6.3 | B635 | PASID is permitted on all Memory Requests, including UIO and DMWr. |
| 6.3 | B640 | <ul style="list-style-type: none"> Correct <u>§ Figure L-4</u> to match <u>§ Table L-2</u> Correct <u>§ Figure L-8</u> to match <u>§ Table L-4</u> |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| | | <ul style="list-style-type: none"> Update both figures to reflect that Sub-Stream is now a 3 bit field. |
| 6.3 | B641 | Increase Retimer Force Timeout value from 48 ms to 96 ms |
| 6.3 | B642 | In 8b/10b and 128b/130b, the <u>Training Control</u> field is bit encoded (e.g., the Hot Reset bit). In 1b/1b, the <u>Training Control</u> field is encoded. This erratum corrects wording elsewhere in the specification to reflect this difference (e.g., use Hot_Reset_Request which is indicated in two different ways based on data rate). |
| 6.3 | B643 | Correct incorporation error in cross reference (there is no Section 6.99.4). |
| 6.3 | B647 | Add VOL parameter to § <u>Table 12-2</u> (PESTI DC Specifications) |
| 6.3 | B648 | Add implementation note Rationale for T2wrst to § <u>Section 12.5.4.2</u> . |
| 6.3 | B648a | In § <u>Figure 12-15</u> , update left end of the arrow for timing parameter $T_{i3c2smb}$. |
| 6.3 | B649 | Update caption of § <u>Table 10-2</u> |
| 6.3 | B650 | Clarify INTx/MSI/MSI-X interrupt wording. |
| 6.3 | B652 | Clarify Precoding behavior in <u>Loopback.Active</u> |
| 6.3 | B653 | Clarify wording for gray coding and precoding of Compliance Pattern in 1b/1b Encoding. |
| 6.3 | B655 | Delete redundant bullet in § <u>Section 4.2.6.7</u> . |
| 6.3 | B657 | Define Flit Mode behavior when receiving TLPs with a Reserved Type encoding. |
| 6.3 | B658 | IDE Fail and IDE Sync messages are not defined in the UIO VC |
| 6.3 | B659 | Clarify Implementation Note <u>TS0 to TS1 Transitions</u> . |
| 6.3 | B661 | NPEM registers are not affected by Function Level Reset |
| 6.3 | B662 | Clarify text indicating that ACS Source Validation does not check Segment Numbers. |
| 6.3 | B665 | PCI-PM is not required for VFs. |
| 6.3 | B666 | Clarify behavior for reserved type encodings in § <u>Table 2-5</u> , Flit Mode TLP Header Type Encodings |
| 6.3 | B667 | Clarify behavior when DOE Object Length is not as expected. |
| 6.3 | B669 | Clarify definition of <u>data payload</u> . |
| 6.3 | B670 | Clarify definition of <u>Physical Function</u> . |
| 6.3 | B671 | Clarify definition of <u>Single-Function Device</u> |
| 6.3 | B672 | Clarify definition of <u>Requester</u> |
| 6.3 | B673 | Clarify software interpretation of the <u>Consecutive Flit Error after the Last Flit Error</u> field. |
| 6.3 | B674 | Refer to tables instead of figures in § <u>Section 2.2.4.2</u> . |
| 6.3 | B675 | Remove outdated text in § <u>Section 6.13</u> |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--------------|--|
| 6.3 | B676 | Correct Poisoned and Nullified wording in § <u>Section 4.2.3.4.1</u> . |
| 6.3 | B677 | Remove PAM4 versions of Lane Margining parameters in § <u>Table 8-13</u> . Move rules into non-PAM4 parameters (i.e., there is only one set of parameters with different behaviors for PAM4 and NRZ). |
| 6.3 | B678 | Clarifications in § <u>Section 2.2.9.2 – Completion Rules for Flit Mode</u> |
| 6.3 | B679 | In § <u>Section 6.6.2</u> , remove redundant text that explicitly lists <u>Link Equalization Request 16.0 GT/s</u> and <u>16.0 GT/s Lane Equalization Control Register</u> . Those registers are covered by the subsequent existing text “All registers in the <u>Physical Layer 16.0 GT/s Extended Capability structure</u> ”. |
| 6.3 | B680 | Correct cross references in § <u>Section 7.7.3.3</u> |
| 6.3 | B681 | Define behavior when <u>Extended Synch</u> is Set and L0p is activating Lanes. |
| 6.3 | B682 | Simplify access rules for MMIO Register Blocks in § <u>Section 6.35</u> . |
| 6.3 | B683 | Clarify <u>Downstream Component Presence</u> behavior. |
| 6.3 | B684 | Clarify Root Complex behavior regarding <u>AtomicOp Requester Enable</u> and <u>AtomicOp Egress Blocking</u> |
| 6.3 | B685 | Indicate which MSI interrupt vector should be used for <u>Link Equalization Request Interrupt Enable</u> . |
| 6.3 | B686 | Correct inconsistent crosslink behavior between Flit Mode and non-Flit Mode |
| 6.3 | B688 | Clarify Ordered Set modification rules for Retimers. |
| 6.3 | B689 | Correct reference in § <u>Table 7-315 – SFI DRS Received</u> description. |
| 6.3 | B690 | Define term <u>Role-Based Error Reporting</u> . Use it in the definition of the <u>Role-Based Error Reporting bit</u> in the <u>Device Capabilities Register</u> . |
| 6.3 | B691 | Define terms <u>DSP</u> and <u>USP</u> . Global replace DP with DSP and UP with USP. |
| 6.3 | B692 | <p>Update <u>§ Figure 2-53</u> to make symbol 6, bit 7 Reserved. Bits 6:0 remain “[For Vendor Definition]”. Only the low 7 bits get copied during translation. This makes it consistent with the existing text:</p> <ul style="list-style-type: none"> The low 7 bits of byte 6 is available for vendor definition. Byte 6, bit 7 is Reserved in Non-Flit Mode and is the EP bit in Flit Mode. <p>Update § <u>Figure 2-56</u> to include the EP bit in symbol 6, bit 7.</p> <p>Update § <u>Figure 2-27</u> through § <u>Figure 2-31</u> to define EP in Symbol 6, bit 7</p> |
| 6.3 | B693 | Use <u>TS0</u> and <u>TS1</u> in text that previously only used <u>TS1</u> . |
| 6.3 | B694 | Clarify L0p rules regarding abandoned requests. |
| 6.3 | B695 | UIO Requests always use 14-bit tags. Clarify conflicting text. |
| 6.3 | B696 | Add OHC-C content to § <u>Figure L-3</u> and § <u>Figure L-7</u> Update Requester Segment value in § <u>Figure L-8</u> . |
| 6.3 | B697 | Endpoint behavior when receiving an ID Routed Message for an unimplemented Function. |
| 6.3 | B698 | Update pseudocode for RX replay buffer controls in § <u>Section 4.2.3.4.2.1.5</u> |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|-------------------------|--|
| 6.3 | B699 | Remove text describing what happens when receiving a TS1 in a case where TS1s can't occur. |
| 6.3 | B701 | Clarify Switch and Root Complex rules regarding support for Flow-Through Selective IDE. |
| 6.3 | B704 | Protocol Multiplexing is not supported in Flit Mode. |
| 6.3 | B705 | Clarify Flit Error Counter interrupt behavior. |
| 6.3 | B705a | Clarify Flit Error Counter Enable affects on Flit Error Counter . |
| 6.3 | B706 | Recomend Lane Numbering behavior in the Flit Logging Extended Capability . |
| 6.3 | B707 | Clarify Type 1 Config Request behavior when Selective IDE Stream are in use. |
| 6.3 | B710 | Clarify discard rules for Link IDE. |
| 6.3 | B713 | Clarify Matching Link and Lane Numbers for 1b/1b Encoding behavior for TS2s in Recovery.RcvrCfg . |
| 6.3 | B714 | In § Figure 4-33 , replace “MAX_UNACKNOWLEDGED_FLITS” with “511” in the two checks so that they read:“(TX_ACKNAK_FLIT_SEQ_NUM – IMPLICIT_RX_FLIT_SEQ_NUM) mod 1023 < 511”. This makes the figure agree with the pseudo code. |
| ↓↑6.4↑ | Errata: Base 6.3 B803△▷ | ↓↑B803↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B807△▷ | ↓↑B807↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B811△▷ | ↓↑B811↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B812△▷ | ↓↑B812↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B814△▷ | ↓↑B814↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B816△▷ | ↓↑B816↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B817△▷ | ↓↑B817↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B818△▷ | ↓↑B818↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B819△▷ | ↓↑B819↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B820△▷ | ↓↑B820↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B821△▷ | ↓↑B821↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B822△▷ | ↓↑B822↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B823△▷ | ↓↑B823↑ |
| ↓↑6.4↑ | Errata: Base 6.3 B824△▷ | ↓↑B824↑ |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|--------------------------------|--------------|--|
| ↑↓6.4↑ Errata: Base 6.3 B825△▷ | ↑↓B825↑ | ↑↓Define and use term Physical Lane Number.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B826△▷ | ↑↓B826↑ | ↑↓Correct wording of Debug Opcodes Requested .↑ |
| ↑↓6.4↑ Errata: Base 6.3 B827△▷ | ↑↓B827↑ | ↑↓Add caution for using L0p at 16.0 GT/s with multiple Retimers.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B828△▷ | ↑↓B828↑ | <ul style="list-style-type: none"> ↑↓Clarity Segment Number rules.↑ ↑↓Removing DPC does not enter Detect with Link is disabled.↑ ↑↓Add default values for SFI Status Register bits.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B829△▷ | ↑↓B829↑ | ↑↓Clarify Hot Reset and Link Disable behavior.↑ |
| ↑↓6.3↑ Errata: Base 6.3 B831△▷ | ↑↓B831↑ | ↑↓Clarify Loopback scrambling and precoding behavior.↑ |
| ↑↓6.3↑ Errata: Base 6.3 B833△▷ | ↑↓B833↑ | ↑↓Add rules for Switch and Root Port Virtual Bridge Behavior in non-D0 State.Previously, these were included by reference from[PCI-to-PCI-Bridge].↑ |
| ↑↓6.3↑ Errata: Base 6.3 B834△▷ | ↑↓B834↑ | ↑↓Include UIO Requests in many places that used to only include Non-Posted Requests.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B836△▷ | ↑↓B836↑ | ↑↓Clarify ACS redirect behavior↑ |
| ↑↓6.4↑ Errata: Base 6.3 B837△▷ | ↑↓B837↑ | ↑↓Clarify Nak Ignore Window and Ack/Nak Latency rules↑ |
| ↑↓6.4↑ Errata: Base 6.3 B839△▷ | ↑↓B839↑ | ↑↓Clarify ACS Source Validation error reporting behavior↑ |
| ↑↓6.4↑ Errata: Base 6.3 B840△▷ | ↑↓B840↑ | ↑↓Recommend behavior to avoid configuring with less than maximum link width↑ |
| ↑↓6.4↑ Errata: Base 6.3 B841△▷ | ↑↓B841↑ | ↑↓Clarify IDE PCRC Check Failed behavior↑ |
| ↑↓6.4↑ Errata: Base 6.3 B842△▷ | ↑↓B842↑ | ↑↓Clarify CREDIT_LIMIT and SHARED_CREDIT_LIMIT update rules.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B843△▷ | ↑↓B843↑ | ↑↓Add Implementation Note: Use Caution When Injecting Too Many Uncorrectable Errors Into A Flit .↑ |
| ↑↓6.4↑ Errata: Base 6.3 B845△▷ | ↑↓B845↑ | ↑↓Clarify SKP Ordered Set insertion rules, especially for L0p.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B858△▷ | ↑↓B858↑ | ↑↓Bytes 8 and 9 of Non-Flit Mode Invalidation Request and Invalidation Completion Messages should be called Destination ID, not Device ID. This affects § Figure 10-18 and § Figure 10-21 . Flit Mode equivalents were already correct.↑ |
| ↑↓6.4↑ Errata: Base 6.3 B859△▷ | ↑↓B859↑ | ↑↓Define Retimer Margin CRC checking rules for 1b/1b.↑ |
| 6.1 | ECN: TDISP | <p>Incorporate TEE Device Interface Security Protocol (TDISP), Approved July 22, 2022</p> <p>Minor issues corrected during TDISP incorporation (not individually marked):</p> |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|--|---|
| | | <ul style="list-style-type: none"> Update reserved field width in § Table 11-25 to account for IS_NON_TEE_MEM Range Attribute. In § Section 11.5.7 , table reference corrected to § Table 11-34 . In § Section 11.2 “An example list of architectural configurations registers that should be locked and tracked are specified is shown in § Section 11.2.6 .” |
| 6.1 | ECN: UIO | Incorporate Unordered I/O <ins>UIO</ins> , Approved March 16, 2023 |
| 6.1 | ECN: Alt-Protocol-DLLP | Incorporate <i>Alternate Protocol DLLP Reservation</i> , Approved December 14, 2022 Section references in Note 3 of § Figure 3-5 were corrected from the published ECN. |
| 6.1 | ECN: DOE 1.1 | Incorporate <i>Data Object Exchange (DOE)</i> , Revision 1.1 Approved September 29, 2022 |
| 6.2 | ECN: 12V-2x6 | Incorporate 12V-2x6 Connector Updates to PCIe Base 6.0 (12V-2x6), <ins>12V-2x6 Connector Updates to PCIe Base 6.0 (12V-2x6),</ins> Approved August 9, 2023 |
| 6.2 | ECN: MMPT | Incorporate Management Message Passthrough via MMIO Mailbox (MMPT), Approved 2023-09-21 Errata to the ECN since publication are also incorporated: <ul style="list-style-type: none"> Added a note that MMB Registers offset 018h-019h are reserved for CXL compatibility. Added reserved bit definitions to the MMB Capabilities Register , the MMB Control Register , and the MMB Status Register where fields are reserved for CXL compatibility. Updated the section numbers to align with PCIe Base Spec v6.1. Updated CXL to [CXL] for notes about legacy compatibility. |
| 6.2 | ECN: Architectural Out-of-Band Management | Incorporate Architectural Out-of-Band Management, Approved 2023-12-12 |
| 6.2 | ECN: Revised CMA-SPDM | Incorporate Component Measurement and Authentication (CMA-SPDM), Revised December 2022 |
| 6.3 | OHC-E Capability Enumeration | Add a three-bit field OHC-E Support in Device Capabilities 3 Register to indicate to software a Function's level of OHC-E support. |
| 6.3 | Removing Prefetchable Terminology | Through misunderstanding and misuse, “Prefetchable” & “Non-Prefetchable” have become damaged words – a source of more trouble than good. This ECN reworks the PCIe Base Specification to more accurately reflect modern device & system requirements, with the ultimate goal of significantly reducing industry confusion and correctly focusing development efforts. |
| 6.3 | NOP Flit Extensions | Defines two new standardized NOP Flit types: <ol style="list-style-type: none"> <u>NOP.Debug</u> (Debug information) <u>NOP.Vendor</u> (Vendor-defined content) Existing NOP Flit definition is renamed as <u>NOP.Empty</u> Flit type. Any NOP Flit type is permitted when a NOP Flit is transmitted. |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description | |
|----------|---|--|---|
| | | <p>NOP.Debug and NOP.Vendor Flits are not subject to retransmission or flow control. Transmission is on a best effort basis.</p> <p>A NOP Flit Counter mechanism is defined to detect missing NOP Flits (in most cases). A NOP Stream ID indicates the origin of the NOP Flit and enables forwarding use cases of NOP Flits.</p> | |
| 6.3 | SNDR _{TX} and R _{LM-TX} Measurement | <p>Tx Signal-to-Noise Distortion Ratio (SNDR) and Ratio of Level Mismatch (R_{LM-TX}) Measurement Methodologies at 64.0 GT/s</p> <p>Describes ways to measure Signal-to-Noise Distortion Ratio (SNDR_{TX}) and Ratio of Level Mismatch (R_{LM-TX}) at 64.0 GT/s using Multi Pulse Response Fit (MPRF)-based methodologies.</p> <p>The new MPRF-based SNDR_{TX} method mitigates the impact of duty cycle error on the SNDR_{TX} measurement.</p> <p>The new MPRF-based R_{LM-TX} method replaces the method where the 61st UI from each consecutive run of 64 or more UIs of the same symbol {0, 1, 2, 3}, is used to establish a mean voltage for the PAM4 signal levels (V₀, V₁, V₂, and V₃) transmitted for PAM4 2-bit symbols for the R_{LM-TX} equation. The new R_{LM-TX} method will be applied on a compliance pattern waveform captured on a real-time oscilloscope. Pulse responses are established based on each PAM4 signal level. The peak of these pulse responses become the input to the R_{LM-TX} expression.</p> <p>The lower data rates are unaffected as SNDR_{TX} and R_{LM-TX} are only defined at 64.0 GT/s.</p> | |
| ↑↑6.4↑ | ECN: Base 6.3 Optical△▷ | ↑↑Optical Aware Retimer↑ | <p>↑↑The ECN provides a retimer-based mechanism to extend PC Express Links partially containing optical technologies. The definition is optical technology neutral so as to avoid bias against any specific optical technology.↑</p> |
| ↑↑6.4↑ | ECN: Base 6.3 PCIe-MI△▷ | ↑↑PCI Express Management Interface (PCIe-MI)↑ | <p>↑↑This ECN builds upon the Chapter 12. Architectural Out-of-Band Management ECN to define an MCTP-based out-of-band management interface for managing PCIe MI Components (e.g., components or adapters). Out-of-band (OOB) management refers to the ability to perform management operations (e.g., inventorying, monitoring, and configuration) on PCIe-MI Components without interacting with software/firmware running on a host CPU. PCIe-MI provides a standardized interface for BMCs to manage PCIe-MI Components over any transport with an MCTP binding (e.g., SMBus/I2C, I3C, PCIe VDM, USB, and MMBI).↑</p> <p>↑↑The PCIe-MI architecture in this ECN provides a framework for out-of-band management of a PCIe-MI Component that is easily extensible to new standardized commands and vendor-specific commands, along with a base set of commands to perform inventorying, monitoring, and configuration operations.↑</p> |
| ↑↑6.4↑ | ECN: Base 6.3 SIOV△▷ | ↑↑Scalable I/O Virtualization (SIOV)↑ | <p>↑↑This ECN introduces Scalable I/O Virtualization (SIOV), a successor to SR-IOV, intended for devices that expose more virtual device interfaces than is currently practical with SR-IOV. A comprehensive description of the use cases and usage model will be found in The Scalable I/O Virtualization project of the Open Compute Project [OCP].↑</p> <p>↑↑Scalable IOV relaxes many of the requirements of SR-IOV, with the expectation that virtualization software (hypervisors, VMMS, container hosting systems, operating systems which map devices into usermode processes) supplies what the SIOV hardware interface does not.↑</p> |
| ↑↑6.4↑ | ECN: Base 6.3 XT△▷ | ↑↑TDISP eXtended TEE (XT) Extensions↑ | <p>↑↑TDISP eXtended TEE (XT) extensions require changes to the following areas of the PCIe base specification:↑</p> <ol style="list-style-type: none"> ↑↑Integrity & Data Encryption (IDE) sections↑ |

Base 6.4 vs Base 6.3

| Revision | Errata / ECN | Description |
|----------|------------------------|--|
| | | <ul style="list-style-type: none"> ◦ ↓↑Introduce a new XT bit in the IDE TLPs and corresponding changes.↑ <p>2. ↓↑Address Translation Services (ATS) sections</p> <ul style="list-style-type: none"> ◦ ↓↑Introduce a new TE bit in the Translation Completions and corresponding changes.↑ <p>3. ↓↑TEE Device Interface Security Protocol (TDISP) sections</p> <ul style="list-style-type: none"> ◦ ↓↑Introduce TLP and security rules associated with the XT/TE bits and corresponding changes.↑ <p>↓↑The following figures were updated to add the XT bit:↑</p> <ul style="list-style-type: none"> • ↓↑§ Figure 2-13↑ • ↓↑§ Figure 6-64↑ • ↓↑§ Figure L-2↑ • ↓↑§ Figure L-3↑ • ↓↑§ Figure L-4↑ • ↓↑§ Figure L-6↑ • ↓↑§ Figure L-7↑ • ↓↑§ Figure L-8↑ |
| ↓↑6.4↑ | ECN: Base 6.3 eSFI△▷ | <p>↓↑System Firmware Intermediary (SFI) functionality was introduced with the Async Hot-Plug Updates ECN against [PCIe-4.0]. It primarily provided:</p> <ul style="list-style-type: none"> • ↓↑Optional functionality for the Firmware-First model, where SFW serves as the intermediary↑ • ↓↑Blocking rogue Config Reads before the device becomes ready following reset↑ • ↓↑SFW intermediary handling of PD, DLL Link Active, Device Readiness Status (DRS)↑ <p>↓↑Hiding & preparatory config of a newly added adapter by SFW, transparent to the OS</p> <p>Enhanced SFI (eSFI) adds the following functionality:</p> <ul style="list-style-type: none"> • ↓↑Control of the SFI Extended Capability structure by the BMC via PCIe-MI, and lock-out of host software control.↑ • ↓↑Advanced traffic blocking/filtering for added protection against malicious devices↑ • ↓↑Triggerable quarantine mode upon device removal to avoid swapping w/o SFW/BMC supervision↑ • ↓↑Making the SFI Configuration Access Mechanism (CAM) optional for RPs↑ |
| ↓↑6.4↑ | ECN: Base 6.3 RC-ECS△▷ | <p>↓↑Enhanced ERR_COR functionality that introduced the ERR_COR Subclass (ECS) field was included in the Async Hot-Plug Updates ECN against [PCIe-4.0]. It defined the ECS field in ERR_COR, which identifies and distinguishes the following subclasses: ECS SIG_SFW for signaling SFW, ECS SIG_OS for signaling OS SW, and ECS Legacy for Requesters that don't support ECS.↑</p> <p>↓↑RC ECS Handling provides "missing" functionality in RP/RCEC AER for handling received ERR_CORs with different ECS field values. Architecting this functionality enables OSs to use ERR_COR concurrently with ERR_COR being used by SFW.↑</p> |

| Revision | Errata / ECN | Description |
|--------------------------------------|--|--|
| ↑↓6.4↑ ECN: Base 6.3 Tx-PS21△▷ | ↑↓Tx PS21 Methodology at 32.0 & 64.0 GT/s↑ | ↑↓Describes a method to measure TX PS21 parameter at 32.0 & 64.0 GT/s more accurately by using brickwall filter. The relationship of Nyquist to 3rd harmonic can vary in different transmitter designs. This was proposed to remove the impact of the 3rd harmonic on the Nyquist measurement.↑ |
| ↑↓6.4↑ ECN: Base 6.3 LTR-MFD△▷ | ↑↓LTR Enhancements for MFD↑ | ↑↓A Multi-Function Device with latency sensitive applications on different Functions can experience interruption or degradation due to an FLR on Function 0. With this ECN, LTR settings for the Device will be maintained without interruption, even after an FLR is performed on any individual function.↑ |

Base 6.4 vs Base 6.3

Base 6.4 vs Base 6.3

Objective of the PCI Express® Architecture

This document defines the “base” specification for the PCI Express architecture, including the electrical, protocol, platform architecture and programming interface elements required to design and build devices and systems. A key goal of the PCI Express architecture is to enable devices from different vendors to inter-operate in an open architecture, spanning multiple market segments including clients, servers, embedded, and communication devices. The architecture provides a flexible framework for product versatility and market differentiation.

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

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

Base 6.4 vs Base 6.3

PCI Express Architecture Specification Organization

The PCI Express specifications are organized as a base specification and a set of companion documents.

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

The companion specifications define a variety of form factors, including mechanical and electrical chapters covering topics including auxiliary signals, power delivery, and the Adapter interconnect electrical budget.

Base 6.4 vs Base 6.3

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 a mixture of capitalization for the remainder.

Numbers and Number Bases

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

All other numbers are decimal.

Implementation Notes

IMPLEMENTATION NOTE:

§

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

Notes

NOTE §

Notes pertain to the specification itself as opposed to implementations of the specification. For example, they are used to describe the document process. They are also used for forward looking information describing anticipated changes for a future version of this specification.

Issues

ISSUE 1

Issues are outstanding items in the specification. They indicate things like missing, outdated, or lower quality artwork, anticipated changes that are being deferred to a subsequent version of the specification, potential errata items noticed during the editing process, etc.

PCI-SIG's goal is to resolve all notes before the 1.0 published release of the specification (where “resolving” includes deferring an item to later, or determining that the item is not needed or is incorrect).

Implementation Notes [blue], Notes [green], and Issues [red] can also be inline.

Base 6.4 vs Base 6.3

Terms and Acronyms

8b/10b

The data encoding scheme¹ used in the PCI Express Physical Layer for 5.0 GT/s and below.

11b/128b/120b

The data encoding scheme used in the PCI Express Physical Layer for 8.0, 16.0, and 32.0 GT/s.

10-Bit Tags

A Tag's capability that provides a total of 10 bits for the Tag field. See Tag.

14-Bit Tags

A Tag's capability that provides a total of 14 bits for the Tag field. See Tag.

Access Control Services , ACS

A set of capabilities and control registers used to implement access control over routing within a PCI Express component.

ACS Violation

An error that applies to a Posted or Non-Posted Request when the Completer detects an access control violation.

Adapter

Used generically to refer to an add-in card or module.

Advanced Error Reporting , AER

Advanced Error Reporting (see § Section 7.8.4).

Alternative Routing-ID , ARI

Alternative Routing-ID Interpretation. Applicable to Requester IDs and Completer IDs as well as Routing IDs.

ARI Device

A Device associated with an Upstream Port, whose Functions each contain an ARI Extended Capability structure.

ARI Downstream Port

A Switch Downstream Port or Root Port that supports ARI Forwarding.

ARI Forwarding

Functionality that enables the Downstream Port immediately above an ARI Device to access the Devices extended Functions. Enabling ARI Forwarding ensures the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.

Asserted

The active logical state of a conceptual or actual signal.

Async Removal

Removal of an adapter or cable from a slot without lock-step synchronization with the operating system (e.g., in an asynchronous manner without button presses).

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

Base 6.4 vs Base 6.3

Atomic Operation , AtomicOp

One of three architected Atomic Operations where a single PCI Express transaction targeting a location in Memory Space reads the location's value, potentially writes a new value to the location, and returns the original value. This read-modify-write sequence to the location is performed atomically. AtomicOps include FetchAdd , Swap , and CAS .

Attribute

Transaction handling preferences indicated by specified Packet header bits and fields (e.g., non-snoop).

Authentication

A process for determining that an entity is what it appears to be (its identity) using defined data objects and digital signatures.

Base Address Register , BAR

Base Address Registers exist within Configuration Space and are used to determine the amount of system memory space needed by a Function and to provide the base address for a mapping to Function memory space. A Base Address Register may map to memory space or I/O space.

Beacon

An optional 30 kHz to 500 MHz in-band signal used to exit the L2 Link Power Management state. One of two defined mechanisms for waking up a Link in L2 (see Wakeup).

BMC

↑↑BMC↑
↑↑Baseboard Management Controller↑

Bridge

One of several defined System Elements. A Function that virtually or actually connects a PCI/PCI-X segment or PCI Express Port with an internal component interconnect or with another PCI/PCI-X bus segment or PCI Express Port. A virtual Bridge in a Root Complex or Switch must use the software configuration interface described in this specification.

by-1 , x1

A Link or Port with one Physical Lane.

by-8 , x8

A Link or Port with eight Physical Lanes.

by-N , xN

A Link or Port with “N” Physical Lanes.

Compare and Swap , CAS

An AtomicOp where the value of a target location is compared to a specified value and, if they match, another specified value is written back to the location. Regardless, the original value of the location is returned.

Character

An 8-bit quantity treated as an atomic entity; a byte.

Clear

A bit is Clear when its value is 0b.

Cold Reset

A Fundamental Reset following the application of main power.

Completer

The Function that terminates or “completes” a given Request, and generates a Completion if appropriate. Generally, the Function targeted by the Request serves as the Completer. For cases when an uncorrectable error prevents the Request from reaching its targeted Function, the Function that detects and handles the error serves as the Completer.

ECN: Base 6.3
eSFI△↔

ECN: Base 6.3
eSFI△↔

Completer Abort , CA

1. A status that applies to a $\downarrow\downarrow$ posted $\downarrow\uparrow$ Posted \uparrow or $\downarrow\downarrow$ non-posted $\downarrow\uparrow$ Non-Posted \uparrow Request that the Completer is permanently unable to complete successfully, due to a violation of the Completer's programming model or to an unrecoverable error associated with the Completer.
2. A status indication returned with a Completion for a $\downarrow\downarrow$ non-posted $\downarrow\uparrow$ Non-Posted Request or a UIO \uparrow Request that suffered a Completer Abort at the Completer.

Errata: Base 6.3
B834△◀▶

Completer ID

The combination of a Completer's Bus Number, Device Number, and Function Number that uniquely identifies the Completer of the Request within a Hierarchy. With an ARI Completer ID, bits traditionally used for the Device Number field are used instead to expand the Function Number field, and the Device Number is implied to be 0.

Completion

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

component

A physical device (a single package).

Configuration Software

The component of system software responsible for accessing Configuration Space and configuring the PCI/PCIe bus.

Configuration Space

One of the four address spaces within the PCI Express architecture. Packets with a Configuration Space address are used to configure Functions.

Configuration-Ready

A Function is Configuration-Ready when it is guaranteed that the Function will respond to a valid Configuration Request targeting the Function with a Completion indicating Successful Completion status.

Containment Error Recovery , CER

A general error containment and recovery approach supported by Downstream Port Containment (DPC), where with suitable software/firmware support, many uncorrectable errors can be handled without disrupting applications.

Conventional PCI

Behaviors or features originally defined in the PCI Local Bus Specification. The PCI Express Base 4.0 and subsequent specifications incorporate the relevant requirements from the PCI Local Bus Specification.

Conventional Reset

A Hot , Warm , or Cold Reset . Distinct from Function Level Reset (FLR).

Data Link Layer

The intermediate Layer that is between the Transaction Layer and the Physical Layer.

Data Link Layer Packet , DLLP

A Packet generated in the Data Link Layer to support Link management functions.

data payload

Information following the header in some packets that is destined for consumption by the targeted Receiver of the Packet $\downarrow\downarrow$ (for example, $\downarrow\uparrow$ e.g., \uparrow Write Requests or Read Completions).

deasserted

The inactive logical state of a conceptual or actual signal.

Deferrable Memory Write , DMWr

A Memory Write where the Requester attempts to write to a given location in Memory Space using the ~~↑↓non posted~~ ↑↓Non-Posted DMWr TLP Type. A Completer that supports this TLP Type can accept or decline the Request, indicating this by means of the Completion status returned. See § [Section 6.32](#).

Design for Testability , DFT

Design for Testability.

Device (uppercase 'D')

A collection of one or more Functions within a single hierarchy identified by common Bus Number and Device Number. An ~~↑↓SR IOV Device~~ ↑↓SR-IOV Device may have additional Functions accessed via additional Bus Numbers and/or Device Numbers configured through one or more SR-IOV Extended Capability structures. ↑↓An SIOV Device may have SDIs accessed via additional Bus Numbers and/or Device Numbers configured through one or more SIOV Extended Capability structures. [↑]

 ECN: Base 6.3
 SIOV $\triangleleft\triangleright$
device (lowercase 'd')

1. A physical or logical entity that performs a specific type of I/O.
2. A component on either end of a PCI Express Link.
3. A common imprecise synonym for Function, particularly when a device has a single Function.

Device Readiness Status , DRS

A mechanism for indicating that a Device is Configuration-Ready (see § [Section 6.22.1](#)).

DLP

In Flit Mode, the Data Link Layer Payload within a Flit.

Downstream

1. The relative position of an interconnect/System Element (Port/component) that is farther from the Root Complex. The Ports on a Switch that are not the Upstream Port are Downstream Ports. All Ports on a Root Complex are Downstream Ports. The Downstream component on a Link is the component farther from the Root Complex.
2. A direction of information flow where the information is flowing away from the Root Complex.

Downstream Path

The flow of data through a Retimer from the Upstream Pseudo Port Receiver to the Downstream Pseudo Port Transmitter.

Downstream Port Containment , DPC

The automatic disabling of the Link below a Downstream Port following an uncorrectable error, which prevents TLPs subsequent to the error from propagating Upstream or Downstream.

~~↑↓DSP~~ ↑↓DSP

Downstream Port

DWORD , DW

Four bytes. Used in the context of a data payload, the 4 bytes of data must be on a naturally aligned 4-byte boundary (the least significant 2 bits of the byte address are 00b).

Egress Port

The transmitting Port; that is, the Port that sends outgoing traffic.

Electrical Idle

A Link state used in a variety of defined cases, with specific requirements defined for the Transmitter and Receiver.

End-End TLP Prefix

A TLP Prefix that is carried along with a TLP from source to destination. See § Section 2.2.10.4 .

Endpoint

One of several defined System Elements. A Function that has a Type 00h Configuration Space header.

Endpoint Upstream Port

An Upstream Port that contains Endpoint Functions exclusively.

↑↑Enhanced SFI , eSFI↑

↑↑A defined set of enhancements for SFI , primarily providing improved security and improved support for a BMC serving as the intermediary. Support for Enhanced SFI is indicated by the Enhanced SFI Supported bit being Set.↑

ECN: Base 6.3
eSFI△<>

ECN: Base 6.3
eSFI△<>

error detection

Mechanisms that determine that an error exists, either by the first agent to discover the error (e.g., Malformed TLP) or by the recipient of a signaled error (e.g., receiver of a poisoned TLP).

error logging

A detector setting one or more bits in architected registers based on the detection of an error. The detector might be the original discoverer of an error or a recipient of a signaled error.

error reporting

In a broad context, the general notification of errors. In the context of the Device Control register, sending an error Message. In the context of the Root Error Command register, signaling an interrupt as a result of receiving an error Message.

error signaling

One agent notifying another agent of an error either by (1) sending an error Message, (2) sending a Completion with ↑↑UR/CA↑ ↑↑UR/ CA↑ Status, or (3) poisoning a TLP.

Extension Device

A component whose purpose is to extend the physical length of a Link.

Extended Function

Within an ARI Device, a Function whose Function Number is greater than 7. Extended Functions are accessible only after ARI-aware software has enabled ARI Forwarding in the Downstream Port immediately above the ARI Device.

FetchAdd , Fetch and Add

An AtomicOp where the value of a target location is incremented by a specified value using two's complement arithmetic ignoring any carry or overflow, and the result is written back to the location. The original value of the location is returned.

Flit

Flow Control Unit

Flow Control

The method for communicating receive buffer status from a Receiver to a Transmitter to prevent receive buffer overflow and allow Transmitter compliance with ordering rules.

Flow Control Packet , FCP

A DLLP used to send Flow Control information from the Transaction Layer in one component to the Transaction Layer in another component.

Flow-Through

Refers to the behavior, by a Switch or Root Complex supporting peer-to-peer, of passing an IDE TLP associated with a Selective IDE Stream from Ingress Port to Egress Port without modification.

↑↓FRU↑

↑↓Field Replacable Unit↑

FRU Information

Information used to provide inventory and capability information about a board on which the **↑↓FRU Information Device↓** **↑↓FRU Information Device↑** is located.

FRU Information Device

A storage component such as a real or firmware emulated **↑↓EEPROM↓** **↑↓EEPROM that stores FRU Information↑** compatible with components such as the AT **↑↓24C256 that stores FRU Information↓** **↑↓24C256↑**

Function

Within a Device, an addressable entity in Configuration Space associated with a single Function Number. Used to refer to one Function of a Multi-Function Device, or to the only Function in a Single-Function Device. Specifically included are special types of Functions defined in **↑↓\$ Chapter 9.↑**, notably **Physical Functions and Virtual Functions**.

Function Group

Within an ARI Device, a configurable set of Functions that are associated with a single Function Group Number. Function Groups can optionally serve as the basis for VC arbitration or access control between multiple Functions within the ARI Device.

Function Level Reset , FLR

A mechanism for resetting a specific Endpoint Function (see § [Section 6.6.2](#)).

Function Readiness Status , FRS

A mechanism for indicating that a Function is Configuration-Ready (see § [Section 6.22.2](#) **↑↓↓** **↑↓↑**).

Fundamental Reset

A hardware mechanism for setting or returning all Port states to the initial conditions specified in this document (see § [Section 6.6](#)).

↑↓GT/s↓ **↑↓GT/s↑**

The number of encoded bits transferred in a second on a direction of a Lane. Short for Giga Transfers per Second.

header

A set of fields that appear at or near the front of a Packet that contain the information required to determine the characteristics and purpose of the Packet.

Hierarchy

A PCI Express I/O interconnect topology, wherein the Configuration Space addresses, referred to as the tuple of Bus/Device/Function Numbers (or just Bus/Function Numbers, for ARI cases), are unique. These addresses are used for Configuration Request routing, Completion routing, some Message routing, and for other purposes. In some contexts a Hierarchy is also called a Segment, and in Flit Mode, the Segment number is sometimes also included in the ID of a Function.

hierarchy domain

The part of a Hierarchy originating from a single Root Port.

Host Bridge

Part of a Root Complex that connects a host CPU or CPUs to a Hierarchy.

Hot Reset

A reset propagated in-band across a Link using a Physical Layer mechanism.

IDE Partner Port

The remote IDE Terminus for an IDE Stream.

IDE Stream

A Port to Port connection established using the mechanisms defined by Integrity and Data Encryption (IDE) to secure TLP traffic between the two Ports. The connection may be in the form of a Selective IDE Stream, in which case it is possible for IDE TLPs to flow through Switches without affecting their security, or in the form of a Link IDE Stream, in which case the two Ports must be connected without intervening Switches.

IDE Terminus

A Port acting as the originator or ultimate destination for IDE TLPs associated with one or more IDE Streams.

IDE TLP

A TLP associated with an IDE Stream and secured using Integrity and Data Encryption (IDE).

↑↓in-band management

↑↓Management operations (e.g., inventorying, monitoring, and configuration) that are performed on a peripheral subsystem (e.g., a PCIe-MI Component) initiated by software/firmware running on a host CPU. See § Chapter 12. .1

ECN: Base 6.3
PCIe-MI $\triangleleft\triangleright$

ECN: Base 6.3
PCIe-MI $\triangleleft\triangleright$

in-band signaling

A method for signaling events and conditions using the Link between two components, as opposed to the use of separate physical (sideband) signals. All mechanisms defined in this document can be implemented using in-band signaling, although in some form factors sideband signaling may be used instead.

Ingress Port

Receiving Port; that is, the Port that accepts incoming traffic.

Internal Error

An error associated with a PCI Express interface that occurs within a component and which may not be attributable to a packet or event on the PCI Express interface itself or on behalf of transactions initiated on PCI Express.

I/O Space

One of the four address spaces of the PCI Express architecture.

isochronous

Data associated with time-sensitive applications, such as audio or video applications.

invariant

A field of a TLP header or TLP Prefix that contains a value that cannot legally be modified as the TLP flows through the PCI Express fabric.

Lane

A set of differential signal pairs, one pair for transmission and one pair for reception. A by-N Link is composed of N Lanes.

Layer

A unit of distinction applied to this specification to help clarify the behavior of key elements. The use of the term Layer does not imply a specific implementation.

Link

The collection of two Ports and their interconnecting Lanes. A Link is a dual-simplex communications path between two components.

Link IDE Stream

An IDE Stream applied to all TLPs, except those associated with Selective IDE Stream(s), where the two Ports are connected without intervening Switches, although extension devices may be present on the Link.

Link Segment

The collection of a Port and a Pseudo Port or two Pseudo Ports and their interconnecting Lanes. A Link Segment is a dual simplex communications path between a Component and a Retimer or between two Retimers (two Pseudo Ports).

Lightweight Notification , LN

This protocol is now deprecated. It was a lightweight protocol that supported notifications to Endpoints via a hardware mechanism when **↑↓cachelines↓↑↑cache lines↑** of interest were updated.

Local TLP Prefix

A TLP Prefix that is carried along with a TLP on a single Link. See § [Section 2.2.10.2](#).

Logical Bus

The logical connection among a collection of Devices that have the same Bus Number in Configuration Space.

Logical Idle

A period of one or more Symbol Times when no information (TLPs, PMUX Packets, DLLPs, or any special Symbol) is being transmitted or received. Unlike Electrical Idle, during Logical Idle the Idle data Symbol is being transmitted and received.

LTR

Abbreviation for Latency Tolerance Reporting

↑↓Malformed Packet↓↑↑Malformed TLP↑

A TLP that violates specific TLP formation rules as defined in this specification.

Measurement

A process for calculating a cryptographic hash value of firmware or other configuration state, applying a digital signature, and returning this information.

Memory Space

One of the four address spaces of the PCI Express architecture.

Message

A TLP used to communicate information outside of the Memory, I/O, and Configuration Spaces.

Message Signaled Interrupt , MSI/MSI-X

Two similar but separate mechanisms that enable a Function to request service by writing a system-specified DWORD of data to a system-specified address using a Memory Write Request. Compared to MSI, MSI-X supports a larger maximum number of vectors and independent message address and data for each vector.

Message Space

One of the four address spaces of the PCI Express architecture.

MMIO

Memory-mapped I/O space. Synonymous with the term Memory Space.

MPS

Abbreviation for Max_Payload_Size.

Multicast , MC

A feature and associated mechanisms that enable a single Posted Request TLP sent by a source to be distributed to multiple targets.

Multicast Group , MCG

A set of Endpoints that are the target of Multicast TLPs in a particular address range.

Multicast Hit

The determination by a Receiver that a TLP will be handled as a Multicast TLP.

Multicast TLP

A TLP that is potentially distributed to multiple targets, as controlled by Multicast Capability structures in the components through which the TLP travels.

Multicast Window

A region of Memory Space where Posted Request TLPs that target it will be handled as Multicast TLPs.

Multi-Function Device , MFD

A Device that has multiple Functions.

Multi-Root I/O Virtualization , MR-IOV

A Function that supports the MR-IOV capability. See [MR-IOV] for additional information.

MUST@FLIT

MUST@FLIT features are mandatory for components that support Flit Mode   Flit Mode Supported  is Set). MUST@FLIT features are strongly recommended for all other components.

naturally aligned

A data payload with a starting address equal to an integer multiple of a power of two, usually a specific power of two. For example, 64-byte naturally aligned means the least significant 6 bits of the byte address are 00 0000b.

NPEM

Native PCIe Enclosure Management

OBFF

Optimized Buffer Flush/Fill

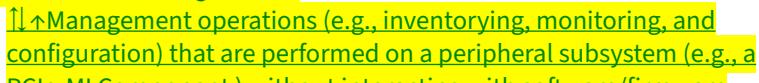
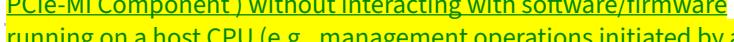
Operating System

Throughout this specification, the terms operating system and system software refer to the combination of power management services, device drivers, user-mode services, and/or kernel mode services.

orderly removal

A hot-plug removal model where the OS is notified when a user/operator wishes to remove an adapter, and the OS has the opportunity to prepare for the event (e.g., quiescing adapter activity) before granting permission for removal.

out-of-band management

  running on a host CPU (e.g., management operations initiated by a BMC). See § Chapter 12..

ECN: Base 6.3
PCIe-MI \triangle 

ECN: Base 6.3
PCIe-MI \triangle 

P2P

Peer-to-peer.

Path

The flow of data through a Retimer, in either the Upstream Path or the Downstream Path.

Packet

A fundamental unit of information transfer consisting of an optional TLP Prefix, followed by a header and, in some cases, followed by a data payload.

Parts per Million , ppm

Applied to frequency, the difference, in millionths of a Hertz, between a stated ideal frequency, and the measured long-term average of a frequency.

↓↓PCIe® PCI Express® ↓ PCI Bridge

See Type 1 Function.

PCI Software Model

The software model necessary to initialize, discover, configure, and use a PCI-compatible device, as specified in [↓↓\[PCI-3.0\]](#), [↑↑\[PCI\],₁](#) [PCI-X], and [Firmware].

↑↑PCIe®↑

[↑↑PCI Express®↑](#)

↑↑PCI Express Management Interface™, PCIe-MI™↑

[↑↑An optional architecture and command set for message-based management of a PCIe-MI Component \(e.g., an adapter or component\) by entities such as in-band management system software running on a host CPU or out-of-band management firmware running on a BMC \(see § Chapter 12. \).](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Command↑

[↑↑A PCIe-MI Message that is transmitted from a PCIe-MI Commander to a PCIe-MI Completer that specifies a management operation to be performed.](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Commander↑

[↑↑An entity on a host \(e.g., system software running on a host CPU or a BMC \) that transmits a PCIe-MI Command to a PCIe-MI Completer.](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Completer↑

[↑↑An entity \(e.g., an MCTP endpoint or a Function supporting MMPT\) on a PCIe-MI Component that processes PCIe-MI Commands and transmits PCIe-MI Completions to a PCIe-MI Commander .](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Completer Reset↑

[↑↑A reset that causes the processing of any PCIe-MI Commands in progress or pending to be aborted and may cause the transmission of any PCIe-MI Completions in progress to be aborted \(see § Section 12.7.1.2 \).](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Completion↑

[↑↑A PCIe-MI Message that is transmitted from a PCIe-MI Completer to the PCIe-MI Commander that transmitted the corresponding PCIe-MI Command in order to provide the results of that PCIe-MI Command .](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Component↑

[↑↑A collection of one or more non-IOV Functions and/or Physical Functions \(e.g., a component or adapter\) that implements the PCI Express Management Interface on one or more PCIe-MI Completers .](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Message↑

[↑↑A PCIe-MI Command or PCIe-MI Completion .](#)

ECN: Base 6.3
PCIe-MI△◀▷

ECN: Base 6.3
PCIe-MI△◀▷

↑↑PCIe-MI Port↑

↑↑A PCIe Upstream Port, PCIe Downstream Port, 2-Wire interface port, or USB port on a PCIe-MI Component.↑

ECN: Base 6.3
PCIe-MI $\triangleleft\triangleright$

ECN: Base 6.3
PCIe-MI $\triangleleft\triangleright$

Phantom Function Number , PFN

An unclaimed Function Number that may be used to expand the number of outstanding transaction identifiers by logically combining the PFN with the Tag identifier to create a unique transaction identifier.

Physical Function , ↑↑PF↑↑PF↑

A Function that contains an SR-IOV Extended Capability structure ↑↑and supports the SR-IOV capabilities.↑↑and/or contains an SIOV Extended Capability structure,↑ defined in ↑↑§ Chapter 9.↑ .

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

Physical Lane

See Lane.

↑↑Physical Lane Number↑

↑↑Lane Number which is assigned in an implementation specific manner and is invariant to Link width and Lane reversal negotiation. Physical lane numbers must be unique and sequential. For example, each Lane of a x16 Link must be assigned a unique Lane number from 0 to 15.↑

Errata: Base 6.3
B825 $\triangleleft\triangleright$

Errata: Base 6.3
B825 $\triangleleft\triangleright$

Physical Layer

The Layer that directly interacts with the communication medium between two components.

Port

1. Logically, an interface between a component and a PCI Express Link.
2. Physically, a group of Transmitters and Receivers located on the same chip that define a Link.

Power Management

Software or Hardware mechanisms used to minimize system power consumption, manage system thermal limits, and maximize system battery life. Power management involves tradeoffs among system speed, noise, battery life, and AC power consumption.

PMUX Channel

A multiplexed channel on a PMUX Link that is configured to transport a specific multiplexed protocol. See § Appendix G. .

PMUX Link

A Link where Protocol Multiplexing is supported and enabled. See § Appendix G. .

PMUX Packet

A non-PCI Express Packet transported over a PCI Express Link. See § Appendix G. .

Precision Time Measurement , PTM

An optional capability for communicating precise timing information between components.

Process Address Space ID , PASID

The Process Address Space ID, in conjunction with the Requester ID, uniquely identifies the address space associated with a transaction.

Programmed I/O , PIO

A transaction sequence that's initiated by a host processor, often as the result of executing a single load or store instruction that targets a special address range, but can be generated by other mechanisms such as the PCI-Compatible Configuration Mechanism. Notably, host processor loads or stores targeting an ECAM address range

generate Configuration Space transactions. Other memory-mapped ranges typically exist to generate Memory Space and I/O Space transactions.

Pseudo Port

1. Logically, an interface between a Retimer and a PCI Express Link Segment.
2. Physically, a group of Transmitters and Receivers located on the same Retimer chip that define a Link Segment.

Quality of Service , QoS

Attributes affecting the bandwidth, latency, jitter, relative priority, etc., for differentiated classes of traffic.

QWORD , QW

Eight bytes. Used in the context of a data payload , the 8 bytes of data must be on a naturally aligned 8-byte boundary (the least significant 3 bits of the address are 000b).

RCiEP

Root Complex Integrated Endpoint.

Read Side Effect

An observable change in system state due to a read. A classic Read Side Effect is a read that returns an element from a FIFO, removing that element from the FIFO. This specification does not define or require the implementation of Read Side Effects.

recommended

Among several possibilities one is particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required.

Receiver , Rx

The component that receives Packet information across a Link.

Receiving Port

In the context of a specific TLP, PMUX Packet, or DLLP, the Port that receives the Packet on a given Link.

Re-driver

A non-protocol aware, software transparent, Extension Device.

Refclk

An abbreviation for Reference Clock.

repeater

An imprecise term for Extension Device.

Reported Error

An error subject to the logging and signaling requirements architecturally defined in this document.

Request

A Packet used to initiate a transaction sequence. A Request includes operation code and, in some cases, address and length, data, or other information.

Requester

The Function or system element that first introduces a transaction sequence into the PCI Express domain.

Requester ID

The combination of a Requester's Bus Number, Device Number, and Function Number that uniquely identifies the Requester within a Hierarchy. With an ARI Requester ID, bits traditionally used for the Device Number field are used instead to expand the Function Number field, and the Device Number is implied to be 0.

Reserved

The contents, states, or information are not defined at this time. Using any Reserved area ↑↓(for example, ↓↑(e.g.,↑ packet header bit-fields, configuration register bits) is not permitted. Reserved register fields must be ↓↓read-only↓ ↑read-only↑ and must return 0 (all 0's for multi-bit fields) when read. Reserved encodings for register and packet fields must not be used. Any implementation dependence on a Reserved field value or encoding will result in an implementation that is not PCI Express-compliant. The functionality of such an implementation cannot be guaranteed in this or any future revision of this specification.

Retimer

A Physical Layer protocol aware, software transparent, Extension Device that forms two separate electrical Link Segments.

Role-Based Error Reporting

A set of error handling semantics based on the role of the agent that detects the error, the type of TLP involved in the error, and the error handling settings of the agent. See [Implementation Note: Use of ERR_COR, ERR_NONFATAL, and ERR_FATAL](#).

Root Complex , RC

A defined System Element that includes at least one Host Bridge, Root Port, or Root Complex Integrated Endpoint.

Root Complex Component

A logical aggregation of Root Ports, Root Complex Register Blocks, Root Complex Integrated Endpoints, and Root Complex Event Collectors.

Root Port , RP

A PCI Express Port on a Root Complex that maps a portion of a Hierarchy through an associated virtual PCI-PCI Bridge.

Routing Element

A term referring to a Root Complex, Switch, or Bridge in regard to its ability to route, multicast, or block TLPs.

Routing ID , RID

Either the Requester ID or Completer ID that identifies a PCI Express ↑↓Function.↓ ↑Function or an SIOV SDI .↑

 ECN: Base 6.3
 SIOV△↔
RP PIO

Root Port Programmed I/O. See [§ Section 6.2.11.3](#).

Rx_MPS_Limit

The computed data payload size limit for a Function receiving a TLP, which is determined by the [Rx_MPS_Fixed](#) bit value and [Max_Payload_Size](#) setting in one or more Functions. See [§ Section 2.2.2](#) for details.

↑↓Scalable IOV , SIOV↑

↑↓A strategy for device virtualization relying on the Virtualization Intermediary (VI) to supply many aspects of a virtual device interface, relying on hardware for data-path operations and software for setup, teardown, error handling in order to simplify each device interface.↑

 ECN: Base 6.3
 SIOV△↔

 ECN: Base 6.3
 SIOV△↔
↑↓Scalable Device Interface , SDI↑

↑↓A device interface exposed by a Physical Function employing Scalable IOV . It has a unique Routing ID , though no Configuration Space or BARs.↑

 ECN: Base 6.3
 SIOV△↔

 ECN: Base 6.3
 SIOV△↔
Segment

See [Hierarchy](#)

Selective IDE Stream

An IDE Stream applied selectively to TLPs based on ranges of Memory Addresses and RIDs, and where it is possible for secured TLPs to flow through Switches without affecting their security.

Set

A bit is Set when its value is 1b.

~~↓↑Shadow Function↓~~ ↑Shadow Function↑

An otherwise unimplemented Function, where its Transaction ID space is used by a Function that implements the Shadow Functions Extended Capability structure.

sideband signaling

A method for signaling events and conditions using physical signals separate from the signals forming the Link between two components. All mechanisms defined in this document can be implemented using in-band signaling, although in some form factors sideband signaling may be used instead.

Single-Function Device , SFD

A Device that has a single Function.

Single Root I/O Virtualization , SR-IOV

A ~~↓↑Function that supports↓~~ ↑strategy for device virtualization relying on↑ the ~~↓↑SR-IOV~~ Extended Capability defined in this specification.↑ ↑device hardware or firmware to supply both control-path and data-path aspects of a virtual device interface.↑

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

Single Root PCI Manager , SR-PCIM

Software responsible for configuration and management of the SR-IOV Extended Capability and ~~↓↑PF/VF↓~~ ↑PF/
VF, the SIOV Extended Capability and PF/SDI,↑ as well as dealing with associated error handling. Multiple implementation options exist; therefore, SR-PCIM implementation is outside the scope of this specification.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

~~↑↓SIOV Device↑~~

~~↓↑A Device containing one or more Functions that have an SIOV Extended Capability structure.↑~~

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

SR-IOV Device

A Device containing one or more Functions that have an SR-IOV Extended Capability structure.

SSD

Solid State Drive

strongly recommended

This item is not mandatory, but in the absence of compelling reasons, it is the most desired choice.

Swap , Unconditional Swap

An AtomicOp where a specified value is written to a target location, and the original value of the location is returned.

Switch

A defined System Element that connects two or more Ports to allow Packets to be routed from one Port to another. To configuration software, a Switch appears as a collection of virtual PCI-to-PCI Bridges [PCI-to-PCI-Bridge].

Switch Port

A Port that contains a Switch Downstream Port Function or at least one Switch Upstream Port Function.

Symbol

A 10-bit quantity when using 8b/10b encoding. An 8-bit quantity when using 128b/130b ~~↓↑encoding↓~~ ↑or 1b/1b encodings.↑

Symbol Time

The period of time required to place a Symbol on a Lane (10 times the $\downarrow\uparrow$ Unit Interval \downarrow $\uparrow\downarrow$ Unit Interval \uparrow when using 8b/10b encoding, 8 times the $\downarrow\uparrow$ Unit Interval \downarrow $\uparrow\downarrow$ Unit Interval \uparrow when using 128b/130b encoding, and 4 times the $\downarrow\downarrow$ Unit Interval \downarrow $\uparrow\downarrow$ Unit Interval \uparrow when using 1b/1b encoding).

System Element

A defined Device or collection of Devices that operate according to distinct sets of rules. The following System Elements are defined: Root Complex, Endpoint, Switch, and Bridge.

System Image , SI

A software component running on a virtual system to which specific Functions, PFs , and VFs can be assigned. Specification of the behavior and architecture of an SI is outside the scope of this specification. Examples of SIs include guest operating systems and shared/non-shared protected domain device drivers.

System Software

Includes System Firmware (BIOS, UEFI), Operating System, VMM, management software, platform vendor's add-on to the Operating System.

Tag

A number assigned to a given Non-Posted Request $\uparrow\downarrow$ or UIO Request \uparrow to distinguish Completions for that Request from other Requests.

 Errata: Base 6.3
B834 Δ \triangleleft
TEE Device Interface (TDI)

The unit of assignment for an IO-virtualization capable device. For example, a TDI may be an entire Device, a non-IOV Function, a PF (and possibly its subordinate $\downarrow\downarrow$ VFs \downarrow , $\uparrow\downarrow$ VFs \uparrow or $\uparrow\downarrow$ SDIs \downarrow , a $\downarrow\downarrow$ VF. \downarrow $\uparrow\downarrow$ VF, or an SDI . \uparrow)

 ECN: Base 6.3
SIOV Δ \triangleleft
TEE-I/O

A conceptual framework for establishing and managing Trusted Execution Environments (TEEs) that include a composition of resources from one or more devices (see § Chapter 11.).

TLP Prefix

Additional information that may be optionally prepended to a TLP. TLP Prefixes are either Local or End-End. A TLP can have multiple TLP Prefixes. See § Section 2.2.10 .

TPH

Abbreviation for TLP Processing Hints

Transaction Descriptor

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

Transaction ID

A component of the Transaction Descriptor including Requester ID and Tag .

Transaction Layer

The Layer that operates at the level of transactions (for example, read, write).

Transaction Layer Packet , TLP

A Packet generated in the Transaction Layer to convey a Request or Completion.

transaction sequence

A single Request and zero or more Completions associated with carrying out a single logical transfer by a Requester.

Transceiver

The physical Transmitter and Receiver pair on a single chip.

Translated Request

A Request using a Translated Memory Address, as indicated by the AT field.

Transmitter , Tx

The component sending Packet information across a Link.

Transmitting Port

In the context of a specific TLP, PMUX Packet, or DLLP, the Port that transmits the Packet on a given Link.

Trusted Computing Base (TCB)

A combination of hardware, firmware, and software, responsible for enforcing a security policy. Bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system. By contrast, parts of a computer system outside the TCB shall not be able to create a condition that would allow any more privileges than are granted to them in accordance with the security policy.

Trusted Execution Environment , TEE

Refers to an environment, which may include only a portion of a device, the whole of a device, or a composition of multiple devices, within which some level of “trust” is established, such that operations (including code execution) that occur within this environment are considered “trustworthy”. It is generally the case that one TEE is isolated from other TEEs that are intended to be distinct, and that all TEEs are isolated from untrusted environments.

Trusted Execution Environment Virtual Machine (Trusted Execution Environment VM , TVM)

A Trusted Execution Environment Virtual Machine as defined in the TEE Device Interface Security Protocol (TDISP) reference architecture (see § Chapter 11.).

Tx_MPS_Limit

The computed data payload size limit for a Function transmitting a TLP, which is determined by the Max_Payload_Size setting in one or more Functions. See § Section 2.2.2 for details.

Type 0 Function

Function with a Type 0 Configuration Space Header (see § Section 7.5.1.2).

Type 1 Function

Function with a Type 1 Configuration Space Header (see § Section 7.5.1.3).

Unconditional Swap , Swap

An AtomicOp where a specified value is written to a target location, and the original value of the location is returned.

Unit Interval , UI

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

↑↓Unordered I/O, UIO↓ ↑↑Unordered I/O , UIO↑

A set of Request/Completion Types used with supporting Virtual Channel(s) to support a Requester-managed ordering model. See § Section 6.34 .

Unsupported Request , UR

1. A status that applies to a ↑↓posted↓ ↑↑Posted↑ or ↑↓non-posted↓ ↑↑Non-Posted↑ Request that specifies some action or access to some space that is not supported by the Completer.
2. A status indication returned with a Completion for a ↑↓non-posted↓ ↑↑Non-Posted Request or a UIO↑ Request that suffered an Unsupported Request at the Completer.

 Errata: Base 6.3
B834△◁

Upstream

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

Upstream Path

The flow of data through a Retimer from the Downstream Pseudo Port Receiver to the Upstream Pseudo Port Transmitter.

***variant***

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

Virtual Function , VF

A Function that is associated with a Physical Function . A VF shares one or more physical resources, such as a Link, with the Physical Function and other VFs that are associated with the same PF.

Virtualization Intermediary , VI

A software component supporting one or more SIs-colloquially known as a hypervisor or virtual machine monitor. Specification of the behavior and architecture of the VI is outside the scope of this specification.

wakeup

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

Warm Reset

A Fundamental Reset without cycling main power.

Zero

The numerical value of zero in a bit, field, or register, of appropriate width for that bit, field, or register.

Base 6.4 vs Base 6.3

Reference Documents

PCI

PCI-3.0

[PCI Local Bus Specification, Revision 3.0](#)

PCIe

↓↑PCIe-6.3↓ ↑↑PCIe-6.4↑

PCI Express Base Specification, Revision [↓↑6.3↓ ↑↑6.4↑](#)

↑↑PCIe-6.3↑

[↑↑PCI Express Base Specification, Revision 6.3↑](#)

PCIe-6.2

[PCI Express Base Specification, Revision 6.2](#)

PCIe-6.1

[PCI Express Base Specification, Revision 6.1](#)

PCIe-6.0.1

[PCI Express Base Specification, Revision 6.0.1](#)

PCIe-6.0

[PCI Express Base Specification, Revision 6.0](#)

PCIe-5.0

[PCI Express Base Specification, Revision 5.0](#)

PCIe-4.0

[PCI Express Base Specification, Revision 4.0](#)

PCIe-3.1

PCIe-3.1a

[PCI Express Base Specification, Revision 3.1a](#)

PCIe-3.0

[PCI Express Base Specification, Revision 3.0](#)

PCIe-2.1

[PCI Express Base Specification, Revision 2.1](#)

PCIe-2.0

[PCI Express Base Specification, Revision 2.0](#)

PCIe-1.1

[PCI Express Base Specification, Revision 1.1](#)

PCIe-1.0

PCIe-1.0a

[PCI Express Base Specification, Revision 1.0a](#)

[PCI Express Card Electromechanical Specification, Revision 6.0 – Work in Progress](#)

Base 6.4 vs Base 6.3

CEM
CEM-5.0
CEM-5.1

[PCI Express Card Electromechanical Specification, Revision 5.1 plus PCI Express CEM Specification, Revision 5.1](#)
[Errata](#)

CEM-4.0

[PCI Express Card Electromechanical Specification, Revision 4.0](#)

CEM-3.0

[PCI Express Card Electromechanical Specification, Revision 3.0](#)

CEM-2.0

[PCI Express Card Electromechanical Specification, Revision 2.0](#)

PCIe-to-PCI-PCI-X-Bridge

[PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0](#)

PCI-to-PCI-Bridge

[PCI-to-PCI Bridge Architecture Specification Revision 1.2](#)

PCI-X

[PCI-X Addendum to the PCI Local Bus Specification, Revision 2.0a](#)

Mini-Card

[PCI Express Mini Card Electromechanical Specification, Revision 2.1](#)

OCuLink

[PCI Express OCuLink Specification, Revision 1.1](#)

M.2

[PCI Express M.2 Specification, Revision 4.0](#)

MR-IOV

[MR-IOV Specification, Revision 1.0](#)

U.2

SFF-8639

[PCI Express SFF-8639 Module Specification, Revision 4.0, Version 1.0](#)

↑↑SNIA SFF-TA-1009↑

[↑↑SNIA SFF-TA-1009 Enterprise and Datacenter Standard Form Factor Pin and Signal Specification](#)

<https://www.snia.org/sff>

Ext-Cabling

[PCI Express External Cabling Specification, Revision 3.0a](#)

ExpressModule

[PCI Express ExpressModule Electromechanical Specification, Revision 1.0](#)

PCI-Hot-Plug

PCI-Hot-Plug-1.1

[PCI Hot-Plug Specification, Revision 1.1](#)

PCI-PM

[PCI Bus Power Management Interface Specification, Revision 1.2](#)

PCI-Code-and-ID

PCI Code and ID Assignment Specification, Revision 1.11 (or later)

Firmware

PCI Firmware Specification, Revision 3.2

↑↑USB↑

↑↑USB-2.0↑

↑↑usb-2.0 Universal Serial Bus Specification, Revision 2.0 <https://www.usb.org>↑

ACPI

Advanced Configuration and Power Interface Specification, Revision 6.2

UEFI

Unified Extensible Firmware Interface (UEFI) Specification, Version 2.8

↓↓EUI-48 EUI-64↓ SMBIOS

↑↑System Management BIOS (SMBIOS) Reference Specification↑ ↓↓System Management BIOS (SMBIOS) Reference Specification,↓ ↑↑ Version 3.6.0 or later

↑↑EUI-48↑

↑↑EUI-64↑

Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)

JEDEC-JESD22-C101

JEDEC JESD22-C101F: Field-Induced Charged-Device Model Test Method for Electrostatic Discharge Withstand Thresholds of Microelectronic Components

JEDEC-JEP155-JEP157

JEDEC JEP155: Recommended ESD Target Levels for HBM/MM Qualification and JEP157 Recommended ESD-CDM Target Levels

ESDA-JEDEC-JS-001-2010

ESDA/JEDEC JS-001-2010: Joint JEDEC/ESDA Standard for Electrostatic Discharge Sensitivity Test – Human Body Model (HBM) – Component Level

ITU-T-Rec-X-667

ITU T-Rec. X.667: Information technology – Procedures for the operation of object identifier registration authorities: Generation of universally unique identifiers and their use in object identifiers

ISO-IEC-9834-8

ISO/IEC 9834-8: Information technology -- Procedures for the operation of object identifier registration authorities -- Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers

RFC-4122

IETF RFC-4122: A Universally Unique IDentifier (UUID) URN Namespace

DNS**RFC-1034**

IETF RFC-1034: DOMAIN NAMES – CONCEPTS AND FACILITIES

PICMG

PICMG

PLUG-PLAY-ISA-1.0a

Plug and Play ISA Specification, Version 1.0a, May 5, 1994

PC-Card[PC-Card](#)**MCTP-VDM****DSP0238**

Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification – <https://www.dmtf.org/dsp/DSP0238>.

SPDM**DSP0274**

DMTF Security Protocol & Data Model (SPDM) Specification – <https://www.dmtf.org/dsp/DSP0274>. IDE_KM requires SPDM Version 1.1 or above. TDISP requires version 1.2 or above.

SPDM-MCTP**DSP0275**

Security Protocol and Data Model (SPDM) over MCTP Binding Specification – <https://www.dmtf.org/dsp/DSP0275>.

AES-GCM

NIST Special Publication 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC – <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>

Secured SPDM**DSP0277**

Secured Messages using SPDM Specification (IDE requires version 1.0 or above) – <https://www.dmtf.org/dsp/DSP0277>

Secured MCTP**DSP0276**

Secured Messages using SPDM over MCTP Binding Specification (version 1.0 or above) – <https://www.dmtf.org/dsp/DSP0276>

CXL**CXL-3.0**

Compute Express Link   [Specification](#) – <https://www.computeexpresslink.org>

SMBus**SMBus Version 3.2**

SMBus Specification – <http://smbus.org/specs/>

I2C

I2C Specification – <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>

I3C**I3C-Basic****I3C-Basic-1.1.1**

MIPI I3C Basic Specification – <https://resources.mipi.org/mipi-i3c-basic-download>

I3C-DCR

I3C Device Characteristic Register – https://www.mipi.org/mipi_i3c_device_characteristics_register

NVMe**NVM-Express**

NVM Express Specification – <https://nvmeexpress.org/specifications>

NVMe-MI

NVM Express Management Interface Specification – <https://nvmeexpress.org/specification/nvme-mi-specification>

Base 6.4 vs Base 6.3

PLDM-Firmware-Update**DSP0267**

PLDM for Firmware Update Specification – <https://www.dmtf.org/dsp/DSP0267>), or

PLDM-Platform-Monitoring-Control**DSP0248**

PLDM for Platform Monitoring and Control Specification – <https://www.dmtf.org/dsp/DSP0248>

MMBI**DSP0282**

Memory-Mapped BMC Interface (MMBI) Specification – <https://www.dmtf.org/dsp/DSP0282>

↑↑↑DMTF-DSP0233↑

[↑↑↑MCTP I3C Binding Specification https://www.dmtf.org/dsp/DSP0233↑](https://www.dmtf.org/dsp/DSP0233)

↑↑↑DMTF-DSP0236↑

[↑↑↑MCTP Base Specification https://www.dmtf.org/dsp/DSP0236↑](https://www.dmtf.org/dsp/DSP0236)

↑↑↑DMTF-DSP0237↑

[↑↑↑MCTP SMBus/I2C Binding Specification https://www.dmtf.org/dsp/DSP0237↑](https://www.dmtf.org/dsp/DSP0237)

↑↑↑DMTF-DSP0256↑

[↑↑↑MCTP Host Interface Specification https://www.dmtf.org/dsp/DSP0256↑](https://www.dmtf.org/dsp/DSP0256)

↑↑↑DMTF-DSP0283↑

[↑↑↑MCTP USB Binding Specification https://www.dmtf.org/dsp/DSP0283↑](https://www.dmtf.org/dsp/DSP0283)

↑↑↑DMTF-DSP0284↑

[↑↑↑MCTP MMBI Binding Specification https://www.dmtf.org/dsp/DSP0284↑](https://www.dmtf.org/dsp/DSP0284)

↑↑↑DMTF-DSP0291↑

[↑↑↑PCIe-MI over MCTP Binding Specification https://www.dmtf.org/dsp/DSP0291↑](https://www.dmtf.org/dsp/DSP0291)

↑↑↑DMTF-DSP2015↑**IPMI-FRU**

IPMI Platform Management FRU Information Storage Definition – <https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/ipmi-platform-mgt-fru-info-storage-def-v1-0-rev-1-3-spec-update.pdf>

CopperLink ↑↑↑TM↑**CopperLink-Internal ↑↑↑TM↑**

CopperLink Internal Cable Specification for PCI Express 5.0 and 6.0 – Work in Progress: Draft 0.9 is <https://members.pcisig.com/wg/PCI-SIG/document/20254>

↑↑↑Open Compute Project↑

[↑↑↑The OCP Community forges new technology norms, fostering an ecosystem where industry players collaborate in a safe framework, shaping a versatile and diverse supply chain. See https://www.opencompute.org↑](https://www.opencompute.org)

↑↑↑OCP Attestation of System Components v1.0 Requirements and Recommendations↑

[↑↑↑https://www.opencompute.org/documents/attestation-v1-0-20201104-pdf↑](https://www.opencompute.org/documents/attestation-v1-0-20201104-pdf)

↑↑OCP Server/MHS↑

↑↑OCP DC/MHS↑

↑↑Open Compute Project Server Modular Hardware System (MHS) Subproject [https://www.opencompute.org/wik.../Server/DC-MHS](https://www.opencompute.org/wiki/Server/DC-MHS)↑

↑↑TCG Reference Integrity Manifest (RIM) Information Model↑

↑↑<https://trustedcomputinggroup.org/resource/tcg-reference-integrity-manifest-rim-information-model>↑

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 attributes, such as usage model, load-store architecture, and software interfaces, are maintained from PCI Local Bus, whereas PCI Local Bus's parallel bus implementation is replaced by a highly scalable, fully serial interface. PCI Express takes advantage of 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 An Evolving I/O Interconnect §

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

- **Supports multiple market segments and emerging applications:**
 - Unifying I/O architecture for desktop, mobile, workstation, server, communications platforms, and embedded devices
- **Ability to deliver low cost, high volume solutions:**
 - Cost at or below PCI cost structure at the system level
- **Support multiple platform interconnect usages:**
 - Chip-to-chip, board-to-board via connector or cabling
- **A variety of mechanical form factors:**
 - [M.2], ~~↓↑[CEM] (Card Electro Mechanical), ↓↑[CEM]~~, [U.2], [OCuLink]
- **PCI-compatible software model:**
 - Ability to enumerate and configure PCI Express hardware using PCI system configuration software implementations with no modifications
 - Ability to boot existing operating systems with no modifications
 - Ability to support existing I/O device drivers with no modifications
 - Ability to configure/enable new PCI Express functionality by adopting the PCI configuration paradigm
- **Performance:**
 - Low-overhead, low-latency communications to maximize application payload bandwidth and Link efficiency
 - High-bandwidth per pin to minimize pin count per device and connector interface
 - Scalable performance via aggregated Lanes and signaling frequency
- **Advanced features:**
 - Comprehend different data types and ordering rules
 - Power management and budgeting
 - Ability to identify power management capabilities of a Device of a specific Function
 - Ability to transition a Device or Function into a specific power state
 - Ability to receive notification of the current power state of a Device of Function
 - Ability to generate a request to wakeup from a power-off state of the main power supply

- Ability to sequence Device power-up to allow graceful platform policy in power budgeting
- Ability to support differentiated services, i.e., different (QoS)
 - Ability to have dedicated Link resources per QoS data flow to improve fabric efficiency and effective application-level performance in the face of head-of-line blocking
 - Ability to configure fabric QoS arbitration policies within every component
 - Ability to ~~↑↓tag↓~~ ↑↑indicate↑ end-to-end QoS with each packet
 - Ability to create end-to-end isochronous (time-based, injection rate control) solutions
- Hot-Plug support
 - Ability to support existing PCI Hot-Plug solutions
 - Ability to support native Hot-Plug solutions (no sideband signals required)
 - Ability to support async removal
 - Ability to support a unified software model for all form factors
- Data Integrity
 - Ability to support Link-level data integrity for all types of transaction and Data Link packets
 - Ability to support end-to-end data integrity for high availability solutions
- Error handling
 - Ability to support PCI-Compatible error handling
 - Ability to support advanced error reporting and handling to improve fault isolation and recovery solutions
- Process Technology Independence
 - Ability to support different DC common mode voltages at Transmitter and Receiver
- Ease of Testing
 - Ability to test electrical compliance via simple connection to test equipment

1.2 PCI Express Link §

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

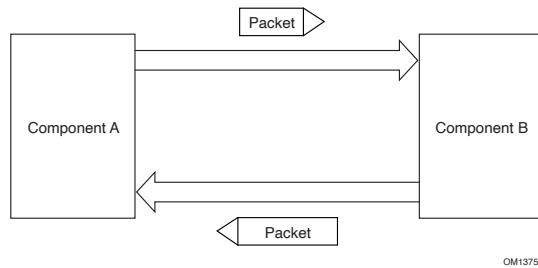


Figure 1-1 PCI Express Link §

The primary Link attributes for PCI Express Link are:

- The basic Link - PCI Express Link consists of dual unidirectional differential Links, implemented as a Transmit pair and a Receive pair. A data clock is embedded using an encoding scheme (see § Chapter 4.) to achieve very high data rates.
- The Signaling method – Each major revision of PCI Express signaling has evolved one (or more) characteristics to increase bandwidth. Throughout this specification, the term **GT/s** is used to refer to the number of encoded bits transferred in a second on a direction of a Lane. The actual effective data rate is dependent on a combination of modulation method, encoding method, and data rate. § Table 1-1 provides a summary of Max Data Rate, Modulation Scheme, Encoding Method, and Effective Max Data Rate with the accounting of only encoding overhead for all the six major revisions of PCI Express.² See § Chapter 4. for more information about the combined signaling method and § Chapter 8. for electrical specification details for each major PCI Express revision.

Table 1-1 PCIe Signaling Characteristics §

| Data Rate | Modulation | Encoding | Effective Data Rate (after removing Encoding overhead) | Base Specification Revision | | | | | |
|-----------|------------|-----------|---|-----------------------------|-----|-----|-----|-----|-----|
| | | | | 6.x | 5.x | 4.x | 3.0 | 2.0 | 1.0 |
| 2.5 GT/s | NRZ | 8b/10b | 2 Gbit/s | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 5.0 GT/s | NRZ | 8b/10b | 4 Gbit/s | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8.0 GT/s | NRZ | 128b/130b | ~8 Gbit/s | ✓ | ✓ | ✓ | ✓ | | |
| 16.0 GT/s | NRZ | 128b/130b | ~16 Gbit/s | ✓ | ✓ | ✓ | | | |
| 32.0 GT/s | NRZ | 128b/130b | ~32 Gbit/s | ✓ | ✓ | | | | |
| 64.0 GT/s | PAM4 | 1b/1b | 64 Gbit/s | ✓ | | | | | |

- Lanes - A Link must support at least one Lane - each Lane represents a set of differential signal pairs (one pair for transmission, one pair for reception). To scale bandwidth, a Link may aggregate multiple Lanes denoted by xN where N may be any of the supported Link widths. A x8 Link operating at the 2.5 GT/s data rate represents an aggregate bandwidth of **↓↑20 Gbit/s/second↓↑↑20 Gbit/s/↑** of raw bandwidth in each direction. This specification describes operations for x1, x2, x4, x8, and x16 Lane widths.
- Data Stream – PCI Express uses Data Stream in Flit Mode and Data Stream in Non-Flit Mode (see § Section 4.2, including § Table 4-1 and § Table 4-2). Support of Data Stream in Non-Flit Mode is mandatory, while support of Data Stream in Flit Mode is mandatory only if a data rate that exceeds 32.0 GT/s is supported.
- Initialization - During hardware initialization, each PCI Express Link is set up following a negotiation of Link width, data rate, and Flit **↓↑mode↓↑↑Mode↑** 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.,↓↑↑(i.e.,↑)** a x16 Link indicates there are 16 differential signal pairs in each **↑↓direction.↓↑↑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 with two Hierarchies composed of a Root Complex (RC), multiple Endpoints, and multiple Switches, interconnected via PCI Express Links.

2. Terms like “PCIe Gen3” are ambiguous and should be avoided. For example, “gen3” could mean (1) compliant with Base 3.0, (2) compliant with Base 3.1 (last revision of 3.x), (3) compliant with Base 3.0 and supporting 8.0 GT/s, (4) compliant with Base 3.0 or later and supporting 8.0 GT/s, **↓.....↓↑.....↑**

Base 6.4 vs Base 6.3

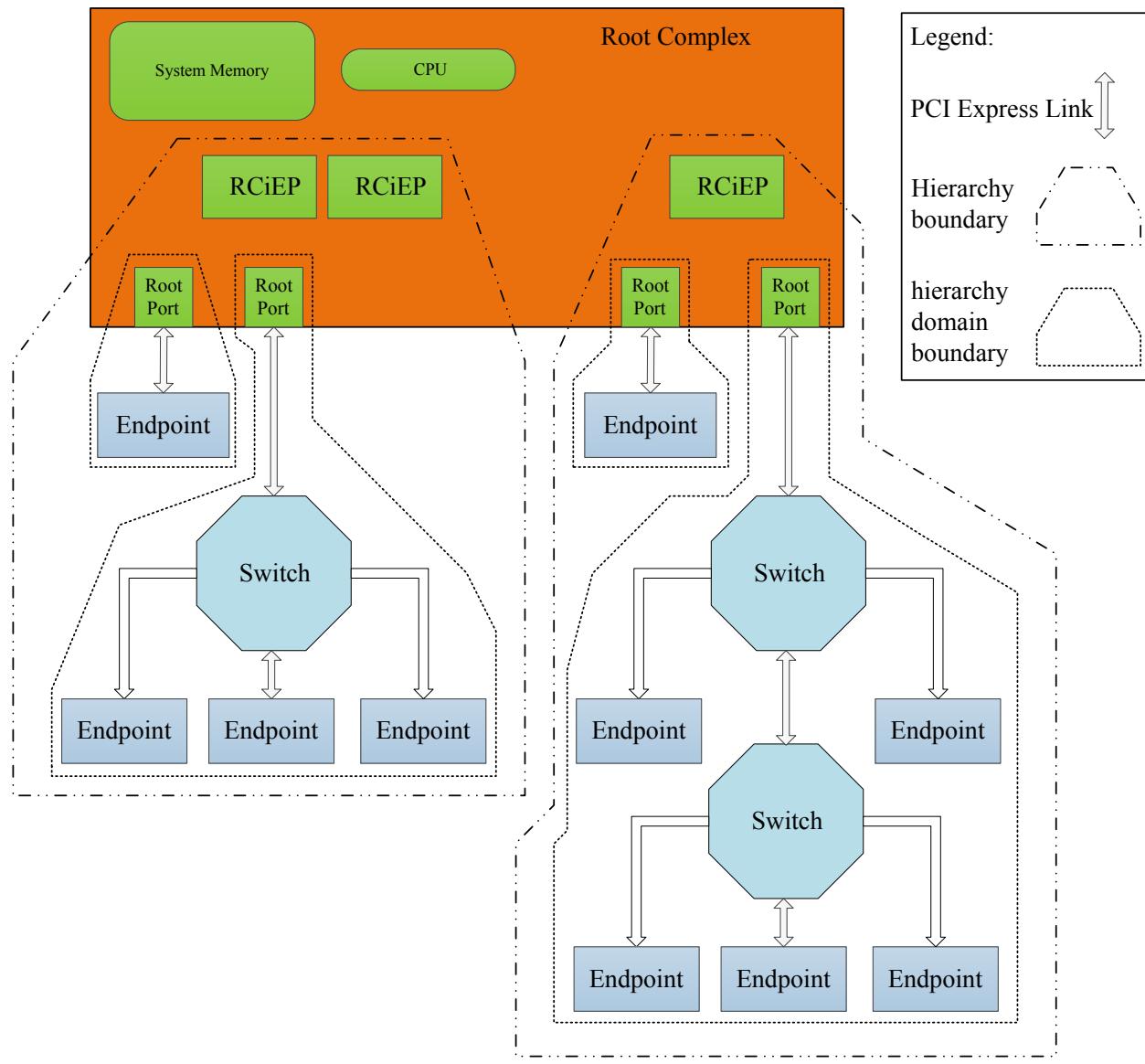


Figure 1-2 Example PCI Express Topology §

1.3.1 Root Complex §

- An RC denotes the root of an I/O hierarchy that connects the CPU/memory subsystem to the I/O.
- As illustrated in § Figure 1-2 , an RC may support one or more PCI Express Ports. Each interface defines a separate hierarchy domain. Each hierarchy domain may be composed of a single Endpoint or a sub-hierarchy containing one or more Switch components and Endpoints.
- The capability to route peer-to-peer transactions between hierarchy domains through an RC is optional and implementation dependent. An RC is permitted to "take ownership" of Requests that pass peer-to-peer between Root Ports, reforming and potentially splitting a Request such that it may appear to the ultimate Completer that the RC was the origin of the Request, and subsequently the RC must reform the Completion(s)

being returned to the original Requester. Alternately, an RC implementation may incorporate a real or virtual Switch internally within the Root Complex to enable full peer-to-peer support in a software transparent way. Unlike the rules for a Switch, an RC is generally permitted to split a packet into smaller packets when routing transactions peer-to-peer between hierarchy domains (except as noted ~~↑↓below~~, e.g., ~~↓↑below~~). For example, an RC is permitted to split a single packet with a 256-byte payload into two packets of 128 bytes payload each. The resulting packets are subject to the normal packet formation rules contained in this specification (e.g., ~~↑↓Max_Payload_Size~~, ~~↓↑Max_Payload_Size~~, ~~↑~~ Read Completion Boundary (RCB), etc.). Component designers should note that splitting a packet into smaller packets may have negative performance consequences, especially for a transaction addressing a device behind a PCI Express to PCI/PCI-X bridge.

Exception: An RC that supports UIO peer-to-peer routing is permitted to split UIO Memory Write Requests only at naturally aligned 64B boundaries.

Exception: An RC that supports peer-to-peer routing of Deferrable Memory Write Requests is not permitted to split a Deferrable Memory Write Request packet into smaller packets (see § Section 6.32).

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

- An RC must support generation of Configuration Requests as a Requester.
- An RC is permitted to support the generation of I/O Requests as a Requester.
 - An RC is permitted to generate I/O Requests to either or both of locations 80h and 84h to a selected Root Port, without regard to that Root Port's PCI Bridge I/O decode configuration; it is recommended that this mechanism only be enabled when specifically needed.
- An RC must not support Lock semantics as a Completer.
- An RC is permitted to support generation of Locked Requests as a Requester.

1.3.2 Endpoints §

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

1.3.2.1 Legacy Endpoint Rules §

- A Legacy Endpoint must be a Function with a Type 00h Configuration Space header.
- A Legacy Endpoint must support Configuration Requests as a Completer.
- A Legacy Endpoint may support I/O Requests as a Completer.
 - A Legacy Endpoint is permitted to accept I/O Requests to either or both of locations 80h and 84h, without regard to that Endpoint's I/O decode configuration.
- A Legacy Endpoint may generate I/O Requests.
- A Legacy Endpoint may support Lock memory semantics as a Completer if that is required by the device's legacy software support requirements.
- A Legacy Endpoint must not issue a Locked Request.
- A Legacy Endpoint may implement Extended Configuration Space Capabilities, but such Capabilities may be ignored by software.

- A Legacy Endpoint operating as the Requester of a Memory Transaction is not required to be capable of generating addresses 4 GB or greater.
- A Legacy Endpoint is required to support MSI or MSI-X or both if an interrupt resource is requested. If MSI is implemented, a Legacy Endpoint is permitted to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure.
- A Legacy Endpoint is permitted to support 32-bit addressing for Base Address Registers that request memory resources.
- A Legacy Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

1.3.2.2 PCI Express Endpoint Rules §

- A PCI Express Endpoint must be a Function with a Type 00h Configuration Space header.
- A PCI Express Endpoint must support Configuration Requests as a Completer.
- A PCI Express Endpoint must not depend on operating system allocation of I/O resources claimed through BAR(s).
- A PCI Express Endpoint must not generate I/O Requests.
- A PCI Express Endpoint must not support Locked Requests as a Completer or generate them as a Requester. PCI Express-compliant device drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.
- A PCI Express Endpoint operating as the Requester of a Memory Transaction is required to be capable of generating addresses greater than 4 GB.
- A PCI Express Endpoint is required to support MSI or MSI-X or both if an interrupt resource is requested. If MSI is implemented, a PCI Express Endpoint must support the 64-bit Message Address version of the MSI Capability structure.
- For generating Memory Requests, 32-bit addressing support is required as described in § Section 2.2.4.1, and 64-bit addressing support is strongly recommended.
- It is strongly recommended for each Memory BAR in a PCI Express component to be 64 bits wide.
- The minimum memory address range requested by a BAR is 128 bytes.
- A PCI Express Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

1.3.2.3 Root Complex Integrated Endpoint Rules §

- A Root Complex Integrated Endpoint (RCiEP) is implemented on internal logic of Root Complexes that contains the Root Ports.
- An RCiEP must be a Function with a Type 00h Configuration Space header.
- An RCiEP must support Configuration Requests as a Completer.
- An RCiEP must not require I/O resources claimed through BAR(s).
- An RCiEP must not generate I/O Requests.
- An RCiEP must not support Locked Requests as a Completer or generate them as a Requester. PCI Express-compliant device drivers and applications must be written to prevent the use of lock semantics when accessing an RCiEP.
- An RCiEP operating as the Requester of a Memory Transaction is required to be capable of generating addresses equal to or greater than the Host is capable of handling as a Completer.

- An RCiEP is required to support MSI or MSI-X or both if an interrupt resource is requested. If MSI is implemented, an RCiEP is permitted to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure.
 - An RCiEP is permitted to support 32-bit addressing for Base Address Registers that request memory resources.
 - An RCiEP must not implement Link Capabilities , Link Status , Link Control , Link Capabilities 2 , Link Status 2 , and Link Control 2 registers in the PCI Express Extended Capability.
 - If an RCiEP is associated with an optional Root Complex Event Collector it must signal PME and error conditions through the Root Complex Event Collector .
 - An RCiEP must not be associated with more than one Root Complex Event Collector .
 - An RCiEP does not implement Active State Power Management .
 - An RCiEP may not be hot-plugged independent of the Root Complex as a whole.
 - An RCiEP must not appear in any of the hierarchy domains exposed by the Root Complex.
 - An RCiEP must not appear in Switches.

1.3.3 Switch §

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

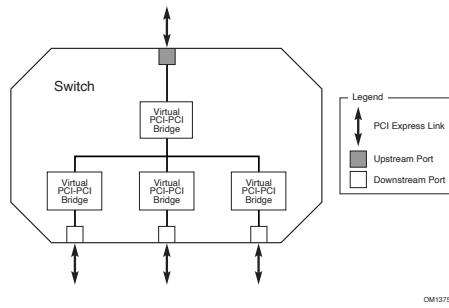
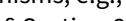
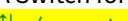


Figure 1-3 Logical Block Diagram of a Switch

- Switches appear to configuration software as two or more logical PCI-to-PCI Bridges.
 - A Switch forwards transactions using PCI Bridge mechanisms; e.g., address-based routing  when engaged in a Multicast, as defined in § Section 6.14 

 - Except as noted in this document, a Switch must forward all types of Transaction Layer Packets (TLPs) between any set of Ports.
 - Locked Requests must be supported as specified in § Section 6.5 . Switches are not required to support Downstream Ports as initiating Ports for Locked Requests.
 - Each enabled Switch Port must comply with the Flow Control specification within this document.
 - A Switch is not allowed to split a packet into smaller   packets. For example,  a single packet with a 256-byte payload must not be divided into two packets of 128 bytes payload each.
 - Arbitration between Ingress Ports (inbound Link) of a Switch may be implemented using round robin or weighted round robin when contention occurs on the same Virtual Channel. This is described in more detail later within the specification.

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

1.3.4 Root Complex Event Collector §

- A Root Complex Event Collector (RCEC) provides support for terminating error and PME messages from RCiEPs.
- A Root Complex Event Collector must follow all rules for an RCiEP (unless otherwise specified).
- A Root Complex Event Collector is not required to decode any memory or I/O resources.
- A Root Complex Event Collector is identified by its Device/Port Type value (see § [Section 7.5.3.2](#)).
- A Root Complex Event Collector has the Base Class 08h, Sub-Class 07h and Programming Interface 00h.³
- A Root Complex Event Collector resides on a Bus in the Root Complex. Multiple Root Complex Event Collectors are permitted to reside on a single Bus.
- A Root Complex Event Collector explicitly declares supported RCiEPs through the [Root Complex Event Collector Endpoint Association Extended Capability](#).
- Root Complex Event Collectors are optional.

1.3.5 PCI Express to PCI/PCI-X Bridge §

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

1.4 Hardware/Software Model for Discovery, Configuration and Operation §

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

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

The PCI Express configuration model supports two mechanisms:

- PCI-compatible configuration mechanism: The PCI-compatible mechanism supports 100% binary compatibility with Conventional PCI aware operating systems and their corresponding bus enumeration and configuration software.

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

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

Each PCI Express Link is mapped through a virtual PCI-to-PCI Bridge structure and has a Logical Bus associated with it. The virtual PCI-to-PCI Bridge structure may be part of a PCI Express Root Complex Port, a Switch Upstream Port, or a Switch Downstream Port. A Root Port is a virtual PCI-to-PCI Bridge structure that originates a PCI Express hierarchy domain from a PCI Express Root Complex. Devices are mapped into Configuration Space such that each will respond to a particular Device Number.

1.5 PCI Express Layering Overview §

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

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

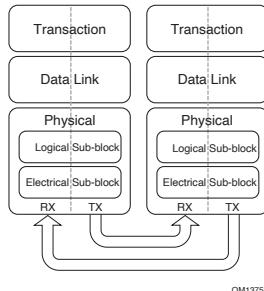


Figure 1-4 High-Level Layering Diagram §

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

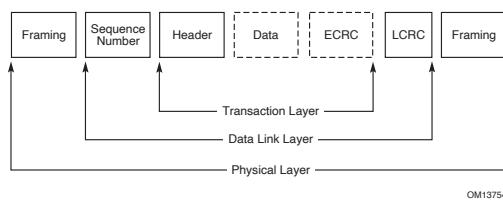


Figure 1-5 Packet Flow Through the Layers §

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

1.5.1 Transaction Layer §

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

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

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

1.5.2 Data Link Layer §

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

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

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

1.5.3 Physical Layer §

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

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

1.5.4 Layer Functions and Services §

1.5.4.1 Transaction Layer Services §

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

Initialization and configuration functions require the Transaction Layer to:

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

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

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

Flow Control services:

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

Ordering rules:

- PCI/PCI-X compliant producer/consumer ordering model,
- Extensions to support Relaxed Ordering,
- Extensions to support ID-Based Ordering,
- Support for UIO ordering model.

Power management services:

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

Virtual Channels and Traffic Class:

- The combination of Virtual Channel mechanism and Traffic Class identification is provided to support differentiated services and QoS support for certain classes of applications. They are also used to provide separate ordering domains for UIO and non-UIO Virtual Channels.

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

1.5.4.2 Data Link Layer Services §

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

Initialization and power management services:

- Accept power state Requests from the Transaction Layer and convey to the Physical Layer
- Convey active/reset/disconnected/power managed state to the Transaction Layer

Data protection, error checking, and retry services:

- CRC generation
- Transmitted TLP storage for Data Link level retry
- Error checking
- TLP acknowledgement and retry Messages
- Error indication for error reporting and logging

1.5.4.3 Physical Layer Services §

Interface initialization, maintenance control, and status tracking:

- Reset/Hot-Plug control/status
- Interconnect power management
- Width and Lane mapping negotiation
- Lane polarity inversion

Symbol and special Ordered Set generation:

- 8b/10b encoding/decoding
- Embedded clock tuning and alignment

Block and special Ordered Set generation:

- 128b/130b encoding/decoding
- 1b/1b encoding/decoding
- Link Equalization

Symbol transmission and alignment:

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

System Design For Testability (DFT) support features:

- Compliance Pattern (see § [Section 4.2.9](#), § [Section 4.2.11](#), and § [Section 4.2.14](#))
- Modified Compliance Pattern (see § [Section 4.2.10](#), § [Section 4.2.12](#), and § [Section 4.2.15](#))
- Jitter Measurement Pattern (see § [Section 4.2.13](#) and § [Section 4.2.16](#))
- Flit Error Injection (see § [Section 7.8.13](#))

1.5.4.4 Inter-Layer Interfaces §

1.5.4.4.1 Transaction/Data Link Interface §

The Transaction to Data Link interface provides:

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

The Data Link to Transaction interface provides:

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

1.5.4.4.2 Data Link/Physical Interface §

The Data Link to Physical interface provides:

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

The Physical to Data Link interface provides:

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

- Actual power state for the Link
- Connection status information

2. Transaction Layer Specification §

2.1 Transaction Layer Overview §

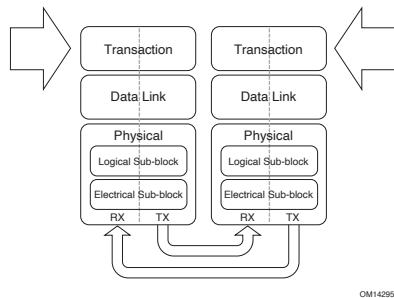


Figure 2-1 Layering Diagram Highlighting the Transaction Layer §

At a high level, the key aspects of the Transaction Layer are:

- A pipelined full Split-Transaction protocol
- Mechanisms for differentiating the ordering and processing requirements of Transaction Layer Packets (TLPs)
- Credit-based flow control
- Optional support for data poisoning and end-to-end data integrity detection.

The Transaction Layer comprehends the following:

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

This chapter specifies the behaviors associated with the Transaction Layer.

2.1.1 Address Spaces, Transaction Types, and Usage §

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

Table 2-1 Transaction Types for Different Address Spaces §

| Address Space | Transaction Types | Basic Usage |
|---------------|---|---|
| Memory | Read Write | Transfer data to/from a memory-mapped location |
| I/O | Read Write | Transfer data to/from an I/O-mapped location |
| Configuration | Read Write | Device Function configuration/setup |
| Message | Baseline (including ↓↓Vendor Defined↓ <ins>↑↑Vendor-Defined↑</ins>) | From event signaling mechanism to general purpose messaging |

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

2.1.1.1 Memory Transactions §

Memory Transactions include the following types:

- Read Request/Completion
- Write Request (and Completions for UIO)
- Deferrable Memory Write Request/Completion
- AtomicOp Request/Completion

Memory Transactions use two different address formats:

- Short Address ~~↓↓Format:↓~~ ↑↑Format↑ 32-bit address
- Long Address ~~↓↓Format:↓~~ ↑↑Format↑ 64-bit address

Certain Memory Transactions can optionally include a ~~↓↓PASID TLP Prefix (Non-Flit Mode)↓~~ ↑↑PASID TLP Prefix (NFM)↑ or ~~↓↓OHC (Flit Mode)↓~~ ↑↑OHC-A1 / OHC-A4 (FM)↑ containing the Process Address Space ID (PASID). See § [Section 6.20](#) for details.

Certain Memory Transactions are required to use only 64-bit address formats.

2.1.1.2 I/O Transactions §

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

- Read Request/Completion
- Write Request/Completion

I/O Transactions use a single address format:

- Short Address ~~↓↓Format:↓~~ ↑↑Format↑ 32-bit address

2.1.1.3 Configuration Transactions §

Configuration Transactions are used to access configuration registers of Functions within devices.

Configuration Transactions include the following types:

- Read Request/Completion
- Write Request/Completion

2.1.1.4 Message Transactions §

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

In addition to specific Messages defined in this document, PCI Express provides support for Vendor-Defined Messages using specified Message codes. Except for Vendor-Defined Messages that use the PCI-SIG® Vendor ID (0001h), the definition of specific Vendor-Defined Messages is outside the scope of this document.

This specification establishes a standard framework within which vendors can specify their own Vendor-Defined Messages tailored to fit the specific requirements of their platforms (see § [Section 2.2.8.6](#)).

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

2.1.2 Packet Format Overview §

Transactions consist of Requests and Completions, which are communicated using packets. [Figure 2-2](#) shows a high level serialized view of a [Non-Flit Mode](#) TLP, consisting of one or more optional TLP Prefixes, a TLP header, a data payload (for some types of packets), and an optional TLP Digest. § [Figure 2-3](#) shows a more detailed view of the TLP. The following sections of this chapter define the detailed structure of the packet headers and digest.

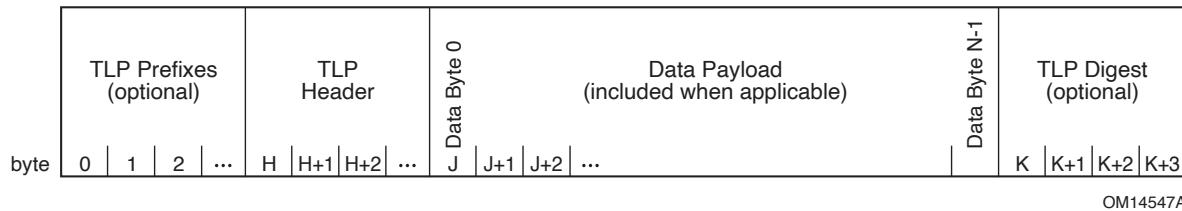


Figure 2-2 Serial View of a [Non-Flit Mode](#) TLP §

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

Detailed layouts of the TLP Prefix, TLP Header and TLP Digest (presented in generic form in § [Figure 2-3](#)) are drawn with the lower numbered bytes on the left rather than on the right as has traditionally been depicted in other PCI specifications. The header layout is optimized for performance on a serialized interconnect, driven by the requirement

that the most time critical information be transferred first. For example, within the TLP header, the most significant byte of the address field is transferred first so that it may be used for early address decode.

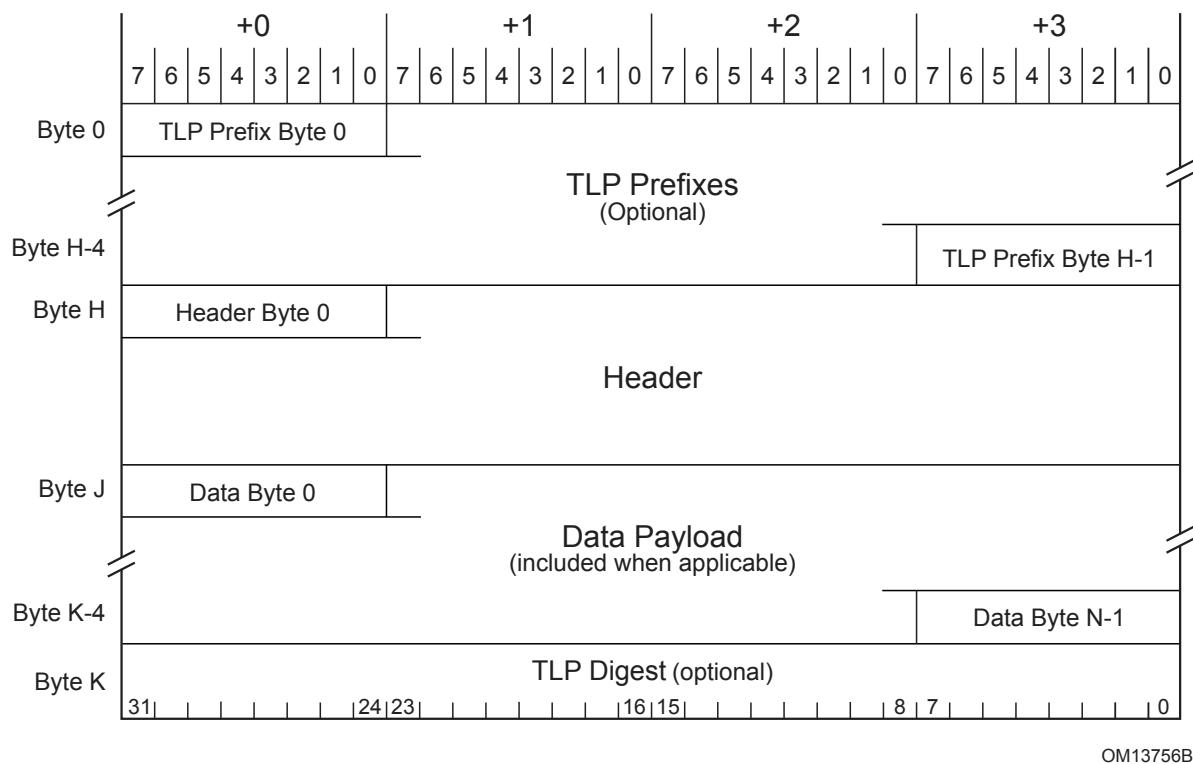


Figure 2-3 Generic TLP Format - Non-Flit Mode [§](#)

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

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

- Format of the packet
- Type of the packet
- Length for any associated data
- Transaction Descriptor, including:
 - Transaction ID
 - Attributes
 - Traffic Class
- Address/routing information
- Byte Enables
- Message encoding
- Completion status

2.2 Transaction Layer Protocol - Packet Definition §

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

A **UIO TLP** is a TLP that is associated with a UIO Virtual Channel.

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. All UIO Requests require Completions. Completions are associated with their corresponding Requests by the value in the Transaction ID field of the Packet header.

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

There are different header formats for Non-Flit Mode (NFM) and Flit Mode (FM). Routing elements must translate between the FM and NFM TLP formats when the Ingress Port and Egress Port are in different modes. In some **cases**, translation is not possible, and the handling of such cases is also defined in this **Chapter**.

2.2.1 Common Packet Header Fields §

2.2.1.1 Common Packet Header Fields for Non-Flit Mode §

All TLP prefixes and headers contain the following fields (see [Figure 2-4](#)):

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

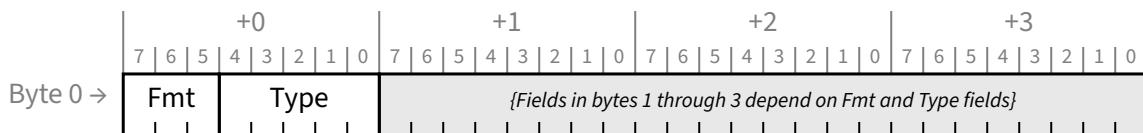


Figure 2-4 Fields Present in All [Non-Flit Mode](#) TLPs §

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

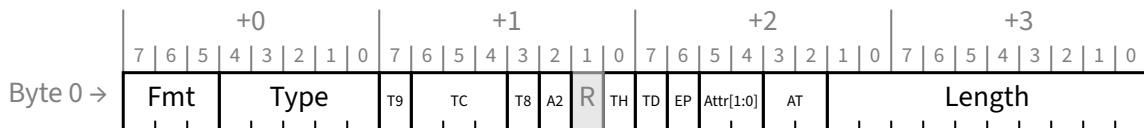
The Fmt and Type fields of the TLP Header provide the information required to determine the size of the remaining part of the TLP Header, and if the packet contains a data payload following the header.

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

Base 6.4 vs Base 6.3

- Permitted Fmt[2:0] and Type[4:0] field values are shown in [Table 2-3](#).
 - All other encodings are Reserved (see [Section 2.3](#)).
- TC[2:0] - Traffic Class (see [Section 2.2.6.6](#)) - bits [6:4] of byte 1
- R (byte 1 bit 1) - Reserved; formerly was the Lightweight Notification (LN) bit, but is now available for reassignment.
- TLP Hints (TH) - 1b indicates the presence of TLP Processing Hints (TPH) in the TLP header and optional TPH TLP Prefix (if present) - bit 0 of byte 1 (see [Section 2.2.7.1.1](#))
- Attr[1:0] - Attributes (see [Section 2.2.6.3](#)) - bits [5:4] of byte 2
- Attr[2] - Attribute (see [Section 2.2.6.3](#)) - bit 2 of byte 1 (shown as A2 in figures)
- TD - 1b indicates presence of TLP Digest in the form of a single Double Word (DW) at the end of the TLP (see [Section 2.2.3](#)) - bit 7 of byte 2
- Error Poisoned (EP) - indicates the TLP is poisoned (see [Section 2.7](#)) - bit 6 of byte 2
- Length[9:0] - Length of data payload, or of data referenced, in DW (see [Table 2-4](#)) - bits 1:0 of byte 2 concatenated with bits 7:0 of byte 3
 - TLP data must be 4-byte naturally aligned and in increments of 4-byte DW.
 - Reserved for TLPs that do not contain or refer to data payloads, including Cpl, CplLk, and Messages (except as specified)

```
↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" } ] }↓
```

Figure 2-5 Fields Present in All Non-Flit Mode TLP Headers [§](#)Table 2-2 Fmt[2:0] Field Values [§](#)

| Fmt[2:0] | Corresponding TLP Format |
|----------|--------------------------|
| 000b | 3 DW header, no data |
| 001b | 4 DW header, no data |
| 010b | 3 DW header, with data |
| 011b | 4 DW header, with data |

| Fmt[2:0] | Corresponding TLP Format |
|----------|--|
| 100b | TLP Prefix |
| | All encodings not shown above are Reserved (see § Section 2.3). |

Table 2-3 *Non-Flit Mode Fmt[2:0] and Type[4:0] Field Encodings* §

| <small>↓↑TLP Type↓ ↓↑Name↑</small> | Fmt [2:0] ⁴ (b) | Type [4:0] (b) | Description |
|--|-------------------------------|---|---|
| MRd | 000 001 | 0 0000 | Memory Read Request |
| MRdLk | 000 001 | 0 0001 | Memory Read Request-Locked |
| MWr | 010 011 | 0 0000 | Memory Write Request |
| IOrd | 000 | 0 0010 | I/O Read Request |
| IOWr | 010 | 0 0010 | I/O Write Request |
| CfgRd0 | 000 | 0 0100 | Type 0 Configuration Read Request |
| CfgWr0 | 010 | 0 0100 | Type 0 Configuration Write Request |
| CfgRd1 | 000 | 0 0101 | Type 1 Configuration Read Request |
| CfgWr1 | 010 | 0 0101 | Type 1 Configuration Write Request |
| TCfgRd | 000 | 1 1011 | Deprecated TLP Type ⁵ |
| DMWr | 010 011 | 1 1011 | Deferrable Memory Write Request ⁶ |
| Msg | 001 | 1 0r ₂ r ₁ r ₀ | Message Request - The sub-field r[2:0] specifies the Message routing mechanism (see § Table 2-20). |
| MsgD | 011 | 1 0r ₂ r ₁ r ₀ | Message Request with data payload - The sub-field r[2:0] specifies the Message routing mechanism (see § Table 2-20). |
| Cpl | 000 | 0 1010 | Completion without Data - Used for I/O, Configuration Write, and Deferrable Memory Write Completions with any Completion Status. Also used for AtomicOp Completions and Read Completions (I/O, Configuration, or Memory) with Completion Status other than Successful Completion. |
| CplD | 010 | 0 1010 | Completion with Data - Used for Memory, I/O, and Configuration Read Completions. Also used for AtomicOp Completions. |
| CplLk | 000 | 0 1011 | Completion for Locked Memory Read without Data - Used only in error case. |
| CplDLk | 010 | 0 1011 | Completion for Locked Memory Read - Otherwise like CplD. |

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

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

6. This TLP Type value was previously used for Trusted Configuration Space (TCS) Writes, which are no longer supported by this specification.

| ↓↑TLP Type↓ ↓↑Name↑ | Fmt [2:0] (b) | Type [4:0] (b) | Description |
|------------------------------------|------------------|---|--|
| FetchAdd | 010 011 | 0 1100 | Fetch and Add AtomicOp Request |
| Swap | 010 011 | 0 1101 | Unconditional Swap AtomicOp Request |
| CAS | 010 011 | 0 1110 | Compare and Swap AtomicOp Request |
| LPrfx | 100 | 0 L ₃ L ₂ L ₁ L ₀ | Local TLP Prefix - The sub-field L[3:0] specifies the Local TLP Prefix type (see § Table 2-38). |
| EPrfx | 100 | 1 E ₃ E ₂ E ₁ E ₀ | End-End TLP Prefix - The sub-field E[3:0] specifies the End-End TLP Prefix type (see § Table 2-39). |
| | | | All encodings not shown above are Reserved (see § Section 2.3). |

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

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

2.2.1.2 Common Packet Header Fields for Flit Mode §

The TLP grammar is defined as:

- zero or more 1DW Local TLP prefixes⁷
- TLP Header Base with size indicated by Type[7:0] field, followed by zero to 7 DW of Orthogonal Header Content (OHC) as indicated by OHC[4:0] field
- TLP data payload of 0 to 1024DW
- TLP Trailer, if present as indicated by TS[2:0] field

It is required to transmit NOP TLPs while TLP transmission is active if there are no other TLPs to transmit. NOP TLPs must be discarded without effect by the Receiver. All header fields other than the Type field are Reserved for NOP TLPs .

Other notable differences between Flit Mode and Non-Flit Mode TLPs include the following:

- Content that in Non-Flit Mode is included in End-to-End TLP prefixes is now incorporated within the header, as **↓↑OHC↓↑↑OHC↑**
- In Flit Mode, Steering Tags are not **↓↑overloaded↓↑↑overlaid↑** with the Byte Enables. The PH, Steering Tags, and AMA/AV fields are consolidated in **↓↑OHC↓↑↑OHC↑**

7. Sequencing requirements between Local Vendor Defined TLP Prefixes and the Flit Mode Local TLP Prefix are implementation specific.

All Flit Mode TLPs contain the same fields in the first DW of the Header Base (see § Figure 2-6  ).

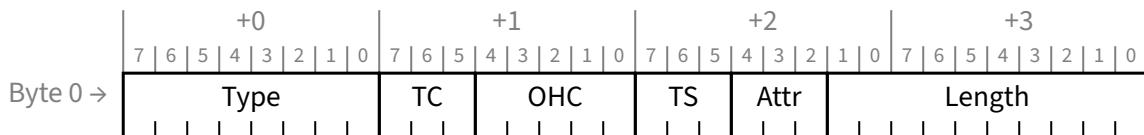


Figure 2-6 First DW of Header Base 

§ Table 2-5 defines the values for the Type[7:0] field for Flit Mode.

- The Type[7:0] field must be fully decoded by all Receivers regardless of which specific encodings are supported.
- All Receivers must handle Flow Control for all Type[7:0] field encodings as specified.
 - For TLPs, where the FC Type is none, Receivers are not required to buffer the TLP, and must silently discard the TLP; for other FC Types:
 - Switch Ports must buffer and route TLPs, including Reserved entries, as specified.
 - Endpoint Upstream Ports and Root Ports are required to buffer, including for Header Logging, up to the largest Header Base size plus all OHC content, but are permitted, after accounting for Flow Control, to discard Header Base and OHC content that is not supported by the Port and not   including   that information in header logging.
- For all Reserved entries, TLP routing must be handled as indicated in the Description field, and the Header Base fields used for routing are at the same location within the Header as with the non-Reserved Header Base formats.
 - Entries marked “Local ... Terminate at Receiver” must be discarded at the Receiving Port.
- A Receiver targeted by a TLP with a Reserved Type[7:0] encoding of FC Type PR or NPR is strongly recommended⁸ to discard the Request following the update of flow control credits, and must handle a TLP with Reserved Type[7:0] encoding of FC Type CPL as an Unexpected Completion.
- A Routing Element that receives a TLP to be forwarded with a Reserved Type[7:0] encoding of FC Type PR or NPR, but is unable to forward it due to a problem like the Egress Port being in DL_Down, is strongly recommended⁹ to discard the Request following the update of flow control credits.
- UIO Requests using FC Type PR are referred to as UIO PR-FC TLPs; UIO Requests using FC Type NPR are referred to as UIO NPR-FC TLPs.

In the Translation Rule column, an entry of “1:1” indicates that there is no change in meaning or behavior when translating between Non-Flit Mode and Flit Mode in either direction. For TLPs that cannot be translated, those not handled by the Ingress Port must be handled by the Egress Port as follows, logging a TLP Translation Egress Blocked error when an error is reported.

  Egress Port errors are handled according to one of the following rules:

-   Egress Rule P → PR FC Type: block at   if TLP is UIO, report no error, else handle as Uncorrectable
-   Egress Rule NP → NPR FC Type: block at   report no error
-   Egress Rule CPL → CPL FC Type: block at   handle as Uncorrectable

8. For backward compatibility with previous versions of this specification, the Request is permitted to be handled as an Unsupported Request.

9. For backward compatibility with previous versions of this specification, the Request is permitted to be handled as an Unsupported Request.

UIO is defined only for FM, and no translation of UIO TLPs to NFM is permitted. UIO TLPs targeting an Egress Port in NFM must be handled as described in the preceding paragraph. Note that error cases involving UIO VC mis-matches are addressed in § Section 2.5.2.

UIO TLPs are indicated as UIO in the Description column. Entries marked Reserved in the description column do not have an assigned VC restriction. A restriction, if required, will be specified when those entries become defined. Entries #0, #141-143 do not have an assigned VC restriction. All other entries are non-UIO TLPs.

Table 2-5 Flit Mode TLP Header Type Encodings §

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|----|------|---|---|---|---|---|---|---|--|------------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | No Operation – Local TLP – Terminate at Receiver | NOP | none | n | 1 | y | NFM uses this Type code for MRd (see #3) |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Memory Read Request Locked, 32b address routed | MRdLk | NPR | n | 3 | n | 1:1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ↓↓IO↓ ↑↑I/O↑ Read Request | IORd | NPR | n | 3 | n | 1:1 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Memory Read Request, 32b address routed | MRd | NPR | n | 3 | y/n | Requires change of Type field value |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Type 0 Configuration Read Request | CfgRd0 | NPR | n | 3 | n | 1:1 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Type 1 Configuration Read Request | CfgRd1 | NPR | n | 3 | n | 1:1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Reserved – ID routed | | CPL | Length | 4 | y | Block at NFM Egress – Uncorrectable |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | CPL | Length | 4 | y | |
| 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Reserved – ID routed | | PR | n | 3 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 9 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | | | PR | n | 3 | y | |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Completion without Data | Cpl | CPL | n | 3 | n | 1:1 |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | Completion without Data, Locked (only) | CplLk | CPL | n | 3 | n | 1:1 |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|-------|------|---|---|---|---|---|---|---|---|-----------------|---------|---------------|-----------------------|-------------------|---------------------------------------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| | | | | | | | | | for error cases) | | | | | | |
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | UIO Write Completion | <i>UIOWrCpl</i> | CPL | n | 3 | y | Block at NFM Egress – Uncorrectable |
| 13 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | UIO Read Completion – No Data | <i>UIORdCpl</i> | CPL | n | 3 | y | Block at NFM Egress – Uncorrectable |
| 14-15 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | X | Reserved – ID routed | | CPL | n | 3 | y | Block at NFM Egress – Uncorrectable |
| 16-19 | 0 | 0 | 0 | 1 | 0 | 0 | X | X | Reserved – 64b address routed | | NPR | Length | 5 | y | Block at NFM Egress – report no error |
| 20-21 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | X | Reserved – 64b address routed | | NPR | Length | 5 | y | Block at NFM Egress – report no error |
| 22-23 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | X | Reserved – 64b address routed | | NPR | Length | 7 | y | Block at NFM Egress – report no error |
| 24 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Reserved – ID routed | | CPL | n | 7 | y | Block at NFM Egress – Uncorrectable |
| 25 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | | | CPL | n | 7 | y | |
| 26 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | | | CPL | Length | 7 | y | |
| 27 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | Reserved – ID routed <i>{was: Trusted Configuration Read (deprecated)}</i> | | CPL | Length | 7 | y | |
| 28-29 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | X | Reserved – ID routed | | NPR | n | 3 | y | Block at NFM Egress – report no error |
| 30-31 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | X | Reserved – ID routed | | NPR | n | 6 | y | Block at NFM Egress – report no error |
| 32 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Memory Read Request, 64b address routed | MRd | NPR | n | 4 | n | 1:1 |
| 33 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | Memory Read Request Locked, 64b | MRdLk | NPR | n | 4 | n | 1:1 |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|-------|------|---|---|---|---|---|---|---|--|---------------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| | | | | | | | | | address routed | | | | | | |
| 34 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | UIO Memory Read Request | <i>UIOMRd</i> | NPR | n | 4 | y | Block at NFM Egress – report no error |
| 35 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | Reserved – 64b address routed | | NPR | n | 4 | y | Block at NFM Egress – report no error |
| 36-39 | 0 | 0 | 1 | 0 | 0 | 1 | X | X | Reserved – ID routed | | NPR | n | 4 | y | Block at NFM Egress – report no error |
| 40-43 | 0 | 0 | 1 | 0 | 1 | 0 | X | X | Reserved – ID routed | | CPL | n | 4 | y | Block at NFM Egress – Uncorrectable |
| 44-45 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | X | Reserved – ID routed | | PR | n | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 46-47 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | X | Reserved – ID routed | | PR | n | 5 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 48 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | Message w/o Data, Routed to Root Complex | Msg | PR | n | 4 | n | 1:1 |
| 49 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Message w/o Data, Routed by Address (64b) - NONE DEFINED | Msg | PR | n | 4 | n | 1:1 |
| 50 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | Message w/o Data, Routed by ID | Msg | PR | n | 4 | n | 1:1 |
| 51 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | Message w/o Data, Broadcast from Root Complex | Msg | PR | n | 4 | n | 1:1 |
| 52 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | Message w/o Data, Local - ↓↓Terminate↓↓ ↑↑Terminate↑↑ at Receiver | Msg | PR | n | 4 | n | 1:1 |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule | |
|-------|------|---|---|---|---|---|---|---|--|-----------|---------|---------------|-----------------------|-------------------|--|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | | |
| 53 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | Message w/o Data, Gathered and routed to RC (PME_TO_Ack) | Msg | PR | n | 4 | n | 1:1 | |
| 54 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | Message w/o Data -- RESERVED | Msg | PR | n | 4 | n | N/A | |
| 55 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | | Msg | PR | n | 4 | n | | |
| 56-59 | 0 | 0 | 1 | 1 | 1 | 0 | X | X | Reserved – 64b address routed | | NPR | n | 4 | y | Block at NFM Egress – report no error | |
| 60-61 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | X | Reserved – ID routed | | NPR | n | 4 | y | | |
| 62-63 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | X | Reserved – ID routed | | NPR | n | 5 | y | | |
| 64 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Memory Write Request, 32b address routed | MWr | PR | Length | 3 | n | 1:1 | |
| 65 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | Reserved – ID routed | | PR | Length | 6 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable | |
| 66 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | ↓↓IO↓ ↑↑I/O↑ Write Request | IOWr | NPR | Length | 3 | n | 1:1 | |
| 67 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | Reserved – ID routed | | PR | Length | 6 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable | |
| 68 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | Type 0 Configuration Write Request | CfgWr0 | NPR | Length | 3 | n | 1:1 | |
| 69 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | Type 1 Configuration Write Request | CfgWr1 | NPR | Length | 3 | n | 1:1 | |
| 70 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | Reserved – ID routed | | NPR | Length | 3 | y | Block at NFM Egress – report no error | |
| 71 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | | | NPR | Length | 3 | y | | |
| 72 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | UIO Read Completion with Data | UIORdCplD | CPL | Length | 3 | y | Block at NFM Egress – Uncorrectable | |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|-------|------|---|---|---|---|---|---|---|---|----------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 73 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | Reserved – ID routed | | CPL | Length | 3 | y | Block at NFM Egress – Uncorrectable |
| 74 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | Completion with Data | CplID | CPL | Length | 3 | n | 1:1 |
| 75 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | Completion with Data, Locked | CplDLk | CPL | Length | 3 | n | 1:1 |
| 76 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | Fetch and Add AtomicOp Request, 32b address routed | FetchAdd | NPR | Length | 3 | n | 1:1 |
| 77 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | Unconditional Swap AtomicOp Request, 32b address routed | Swap | NPR | Length | 3 | n | 1:1 |
| 78 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | Compare and Swap AtomicOp Request, 32b address routed | CAS | NPR | Length | 3 | n | 1:1 |
| 79 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | Reserved – 64b address routed | | PR | n | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 80-83 | 0 | 1 | 0 | 1 | 0 | 0 | X | X | Reserved – 64b address routed | | NPR | Length | 6 | y | Block at NFM Egress – report no error |
| 84-85 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | X | Reserved – 64b address routed | | NPR | Length | 6 | y | |
| 86-87 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | X | Reserved – 64b address routed | | NPR | Length | 7 | y | |
| 88-89 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | X | Reserved – ID routed | | PR | Length | 3 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 90 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | Reserved – 64b address routed | | PR | Length | 4 | y | |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|---------|------|---|---|---|---|---|---|---|---|----------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 91 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | Deferrable Memory Write Request, 32b address routed {was: Trusted Configuration Write (deprecated)} | DMWr | NPR | Length | 3 | n | 1:1 |
| 92-93 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | X | Reserved – ID routed | | PR | Length | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 94-95 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | X | Reserved – ID routed | | PR | Length | 5 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 96 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Memory Write Request, 64b address routed | MWr | PR | Length | 4 | n | 1:1 |
| 97 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | UIO Memory Write Request | UIOMWr | PR | Length | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 98-99 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | X | Reserved – 64b address routed | | PR | Length | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 100-103 | 0 | 1 | 1 | 0 | 0 | 1 | X | X | Reserved – 64b address routed | | PR | Length | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 104-107 | 0 | 1 | 1 | 0 | 1 | 0 | X | X | Reserved – 64b address routed | | PR | Length | 4 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 108 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | Fetch and Add AtomicOp Request, 64b address routed | FetchAdd | NPR | Length | 4 | n | 1:1 |
| 109 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | Unconditional Swap AtomicOp Request, 64b address routed | Swap | NPR | Length | 4 | n | 1:1 |
| 110 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | Compare and Swap AtomicOp Request, 64b address routed | CAS | NPR | Length | 4 | n | 1:1 |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|-----|------|---|---|---|---|---|---|---|---|------|---------|---------------|-----------------------|-------------------|---------------------------------------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 111 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | Reserved – 64b address routed | | NPR | Length | 4 | y | Block at NFM Egress – report no error |
| 112 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Message with Data, Routed to Root Complex | MsgD | PR | Length | 4 | n | 1:1 |
| 113 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | Message with Data, Routed by Address (64b) - NONE DEFINED | MsgD | PR | Length | 4 | n | 1:1 |
| 114 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | Message with Data, Routed by ID | MsgD | PR | Length | 4 | n | 1:1 |
| 115 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | Message with Data, Broadcast from Root Complex | MsgD | PR | Length | 4 | n | 1:1 |
| 116 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | Message with Data, Local - ↑↓terminator↓ ↑↑Terminator↑ at Receiver | MsgD | PR | Length | 4 | n | 1:1 |
| 117 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | Message with Data, Gathered and routed to RC (MsgD NOT USED) | MsgD | PR | Length | 4 | n | 1:1 |
| 118 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | Message with Data -- RESERVED | MsgD | PR | Length | 4 | n | N/A |
| 119 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | RESERVED | MsgD | PR | Length | 4 | n | |
| 120 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | Reserved – 64b address routed | | NPR | Length | 4 | y | Block at NFM Egress – report no error |
| 121 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | | | NPR | Length | 4 | y | |
| 122 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | | | NPR | Length | 4 | y | |
| 123 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | Deferrable Memory Write Request, 64b address routed | DMWr | NPR | Length | 4 | n | 1:1 |

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|---------|------|---|---|---|---|---|---|---|---|-----------------------|---------|---------------|-----------------------|-------------------|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 124-127 | 0 | 1 | 1 | 1 | 1 | X | X | | Reserved – 64b address routed | | NPR | Length | 4 | y | Block at NFM Egress – report no error |
| 128-135 | 1 | 0 | 0 | 0 | 0 | X | X | X | Reserved – Local TLP Prefix – Terminate at Receiver | | none | n | 1 | n | N/A |
| 136-139 | 1 | 0 | 0 | 0 | 1 | 0 | X | X | Reserved – Local TLP Prefix – Terminate at Receiver | <i>FlitModePrefix</i> | none | n | 1 | n | N/A |
| 140 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | Reserved – Local TLP Prefix – Terminate at Receiver | | none | n | 1 | n | N/A |
| 141 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Flit Mode Local TLP Prefix | <i>VendPrefixL0</i> | none | n | 1 | n | N/A |
| 142 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 DW ↓↓Prefix↓ ↑↑Prefix↑↑ Vendor Defined Local 0 | | none | n | 1 | n | N/A |
| 143 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 DW Prefix - Vendor Defined Local 1 | <i>VendPrefixL1</i> | none | n | 1 | n | N/A |
| 144-147 | 1 | 0 | 0 | 1 | 0 | 0 | X | X | Reserved – 64b address routed | | PR | n | 4 | y | Strongly Recommended: Block at NFM Egress / Permitted: Terminate at FM Ingress Port. If UIO TLP report no error, else handle as Uncorrectable ¹⁰ |
| 148-151 | 1 | 0 | 0 | 1 | 0 | 1 | X | X | Reserved – 64b address routed | | PR | n | 5 | y | Strongly Recommended: Block at NFM Egress / Permitted: Terminate at FM Ingress Port. If UIO TLP report no error, else handle as Uncorrectable ¹⁰ |
| 152-155 | 1 | 0 | 0 | 1 | 1 | 0 | X | X | Reserved – 64b address routed | | PR | n | 6 | y | Strongly Recommended: Block at NFM Egress / Permitted: Terminate at FM Ingress Port. If UIO TLP report no error, else handle as Uncorrectable ¹⁰ |
| 156-159 | 1 | 0 | 0 | 1 | 1 | 1 | X | X | Reserved – 64b address routed | | PR | n | 7 | y | Strongly Recommended: Block at NFM Egress / Permitted: Terminate at FM Ingress Port. If UIO TLP report no error, else handle as Uncorrectable ¹⁰ |

10. TLP Translation Egress Blocked at Egress Port, or Unsupported Request at Ingress Port.

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|---------|------|---|---|---|---|---|---|---|-------------------------------|------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| | | | | | | | | | | | | | | | handle as Uncorrectable ¹¹ |
| 160-167 | 1 | 0 | 1 | 0 | 0 | X | X | X | Reserved – 64b address routed | | NPR | n | 5 | y | Block at NFM Egress – report no error |
| 168-169 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | X | Reserved – ID routed | | PR | n | 6 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 170-171 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | X | Reserved – ID routed | | PR | n | 7 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 172-173 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | X | Reserved – ID routed | | CPL | n | 5 | y | Block at NFM Egress – Uncorrectable |
| 174-175 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | X | Reserved – ID routed | | CPL | n | 6 | y | Block at NFM Egress – Uncorrectable |
| 176-183 | 1 | 0 | 1 | 1 | 0 | X | X | X | Reserved – 64b address routed | | PR | Length | 5 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 184-191 | 1 | 0 | 1 | 1 | 1 | X | X | X | Reserved – 64b address routed | | PR | Length | 5 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 192-199 | 1 | 1 | 0 | 0 | 0 | X | X | X | Reserved – 64b address routed | | NPR | n | 6 | y | Block at NFM Egress – report no error |
| 200-201 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | X | Reserved – ID routed | | NPR | n | 7 | y | Block at NFM Egress – report no error |
| 202-203 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | X | Reserved – ID routed | | CPL | Length | 5 | y | Block at NFM Egress – Uncorrectable |
| 204-205 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | X | Reserved – ID routed | | CPL | Length | 6 | y | Block at NFM Egress – Uncorrectable |

11. TLP Translation Egress Blocked at Egress Port, or Unsupported Request at Ingress Port.

Base 6.4 vs Base 6.3

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|---------|------|---|---|---|---|---|---|---|--|------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 206-207 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | X | Reserved – ID routed | | PR | Length | 7 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 208-215 | 1 | 1 | 0 | 1 | 0 | X | X | X | Reserved – 64b address routed | | PR | Length | 6 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 216-223 | 1 | 1 | 0 | 1 | 1 | X | X | X | Reserved – 64b address routed | | PR | Length | 6 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |
| 224-225 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | X | Reserved – Local TLP – Terminate at Receiver | | none | n | 4 | y | N/A |
| 226-227 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | X | Reserved – Local TLP – Terminate at Receiver | | none | n | 6 | y | N/A |
| 228-229 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | X | Reserved – Local TLP – Terminate at Receiver | | none | Length | 4 | y | N/A |
| 230-231 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | X | Reserved – Local TLP – Terminate at Receiver | | none | Length | 6 | y | N/A |
| 232-239 | 1 | 1 | 1 | 0 | 1 | X | X | X | Reserved – 64b address routed | | NPR | n | 7 | y | Block at NFM Egress – report no error |
| 240-241 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | X | Reserved – ID routed | | NPR | Length | 4 | y | Block at NFM Egress – report no error |
| 242-243 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | X | Reserved – ID routed | | NPR | Length | 5 | y | Block at NFM Egress – report no error |
| 244-245 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | X | Reserved – ID routed | | NPR | Length | 6 | y | Block at NFM Egress – report no error |

| # | Type | | | | | | | | Description | Name | FC Type | Data Payload? | Header Base Size (DW) | New for Flit Mode | Translation Rule |
|---------|------|---|---|---|---|---|---|---|-------------------------------|------|---------|---------------|-----------------------|-------------------|--|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
| 246-247 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | X | Reserved – ID routed | | NPR | Length | 7 | y | Block at NFM Egress – report no error |
| 248-255 | 1 | 1 | 1 | 1 | 1 | X | X | X | Reserved – 64b address routed | | PR | Length | 7 | y | Block at NFM Egress – if UIO TLP report no error, else handle as Uncorrectable |

The TS[2:0] field indicates Trailer Size and ~~↑↓use↓↑↑is↑~~ encoded as:

- 000b – No Trailer
- 001b – 1DW Trailer ~~↑↑– Content Reserved if OHC-C is present and the Sub-Stream field indicates an IDE TLP; Else 1DW Trailer↑~~ containing ~~↑↑an↑~~ ECRC ¹².
- 010b – 1DW Trailer – Content Reserved
- 011b – 2DW Trailer – Content Reserved
- 100b – 2DW Trailer – Content Reserved
- 101b – 3DW Trailer with IDE MAC if ~~↑↓and only if↑↑an↑~~ OHC-C ~~↑↑is↑~~ present and ~~↑↑the Sub-Stream field↑~~ indicates ~~↑↑an↑~~ IDE TLP; Else 3DW Trailer – Content Reserved
- 110b – 4DW Trailer with IDE MAC and PCRC if ~~↑↓and only if↑↑an↑~~ OHC-C ~~↑↑is↑~~ present and ~~↑↑the Sub-Stream field↑~~ indicates ~~↑↑an↑~~ IDE TLP; Else 4DW Trailer – Content Reserved
- 111b – 5DW Trailer – Content Reserved

Errata: Base 6.3
B821△◀

The definitions of the TC, Attr and Length fields in Flit Mode are the same as in Non-Flit Mode.

Bit 1 in byte 1 of Non-Flit Mode is now Reserved, but it was the LN bit associated with the now deprecated Lightweight Notification (LN) protocol. This bit is not supported in Flit Mode. Thus, it must be ignored when translating from Non-Flit Mode to Flit Mode, and it must be set to 0b when translating from Flit Mode to Non-Flit Mode.

The OHC[4:0] field indicates the presence of “Orthogonal Header Content” (OHC) encoded as:

- 0 0000b = No OHC present
- x xxx1b = ~~↑↓OHC-A↓↑↑OHC-Ax↑~~ present
- x xx1xb = OHC-B present
- x x1xxb = OHC-C present
- 0 0xxxb = No OHC-E present
- 0 1xxxb = OHC-E1 present
- 1 0xxxb = OHC-E2 present
- 1 1xxxb = OHC-E4 present

12. ~~↑↑Earlier versions of this specification defined this combination as always containing an ECRC, even for IDE TLPs. Receiver behavior is undefined for implementations compliant to those earlier specifications↑~~

When present, OHC must follow the Header Base. It is permitted for any combination of OHC content to be present, but, when present, must follow the Header Base, in A-B-C-E order. The contents of the OHC in some cases varies depending on the TLP type.

For specific TLP types, as defined in this specification, specific OHC content must be included by the Transmitter. Receivers must check for violations of these rules. If a Receiver determines that a Request violates a rule requiring specific OHC content, the Request must be handled as an Unsupported Request. If a Receiver determines that a Completion violates a rule requiring specific OHC content, the Completion must be handled as an Unexpected Completion.

Table 2-6 \downarrow OHC-A \downarrow \uparrow OHC-Ax \uparrow Included Fields for OHC-A1 through OHC-A5 (see § Figure 2-7 through § Figure 2-11) §

| Name | Required for | Byte Enables | PASID, PV | ER, PMR | Destination Segment, DSV | Completer Segment | Completion Status | Lower Address[1:0] | NW Flag |
|--------|--|--------------|-----------|---------|--|-------------------|-------------------|--------------------|---------|
| OHC-A1 | Memory Requests with explicit Byte Enables and/or PASID Address Routed Messages with PASID and Route to Root Complex Messages with PASID Translation Requests | Y | Y | Y | | | | | Y |
| OHC-A2 | \downarrow I/O \downarrow \uparrow I/O \uparrow Requests | Y | | | | | | | |
| OHC-A3 | Configuration Requests | Y | | | Y | | | | |
| OHC-A4 | ID-Routed Messages that require Destination Segment and/or PASID | | Y | | Y | | | | |
| OHC-A5 | Completions when required as defined in § Section 2.2.9.2 | | | | Y | Y | Y | | Y |
| OHC-Ax | Others | | | | When OHC-A is present on other TLPs, all OHC-A bits are Reserved | | | | |

```
 $\uparrow\downarrow$ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 7, "msbyte": 0, "msbit": 7, "name": "NW", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 6, "msbyte": 0, "msbit": 6, "name": "PV", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 5, "name": "PMR", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 4, "name": "ER", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 3, "msbyte": 3, "msbit": 3, "name": "PASID", "attr": "ro" }, { "lsbyte": 3, "lsbit": 4, "msbyte": 3, "msbit": 7, "name": "Last DW BE", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 3, "msbit": 3, "name": "1st DW BE", "addClass": "regFieldSmallText", "attr": "ro" } ] } $\downarrow$ 
```

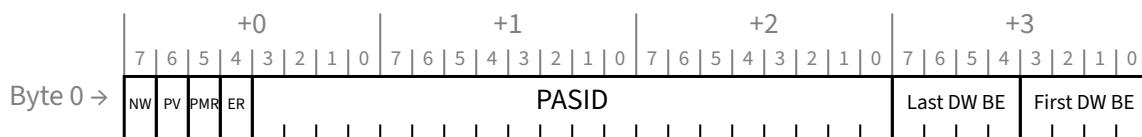


Figure 2-7 OHC-A1 §

Base 6.4 vs Base 6.3

In OHC-A1

- The ER bit is Execute Requested, and the PMR bit is Privileged Mode Requested (see § Section 6.20). These bits are Reserved for all Requests other than Translation Requests (see § Section 10.2.2 ↑↑↑ and Page Requests (see § Section 10.4).
- When OHC-A1 is included with a TLP, if the PASID is not known or has not been assigned, then the PV ("PASID Valid") bit must be Clear.
- The ER and PMR bits are Reserved if PV is Clear.
- The PASID field is Reserved if PV is Clear.
- The NW bit is No ↓↓Write (NW).↓ ↑↑Write.↑ This bit is Reserved for all Requests other than Translation Requests.
- OHC-A1 is required as specified in § Section 2.2.5.2 .

```
↓↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 3, "lsbit": 4, "msbyte": 3, "msbit": 7, "name": "Last DW BE", "addClass": "regFieldSmallText", "attr": "ro" },
    { "lsbyte": 3, "lsbit": 0, "msbyte": 3, "msbit": 3, "name": "1st DW BE", "addClass": "regFieldSmallText", "attr": "ro" } ] }↓
```

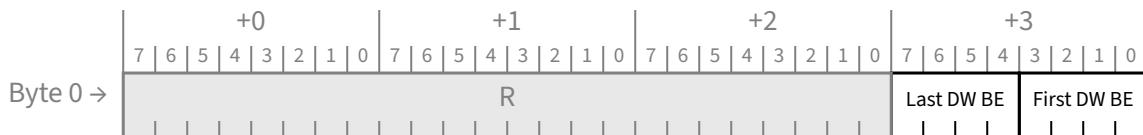


Figure 2-8 OHC-A2 §

In OHC-A2 :

- OHC-A2 is required as specified in § Section 2.2.5.2 .

```
↓↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Destination Segment / Reserved", "addClass": "regFieldVerySmallText", "attr": "ro" },
    { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "DSV", "addClass": "regFieldVerySmallText", "attr": "ro" },
    { "lsbyte": 3, "lsbit": 4, "msbyte": 3, "msbit": 7, "name": "Last DW BE", "addClass": "regFieldSmallText", "attr": "ro" },
    { "lsbyte": 3, "lsbit": 0, "msbyte": 3, "msbit": 3, "name": "1st DW BE", "addClass": "regFieldSmallText", "attr": "ro" } ] }↓
```

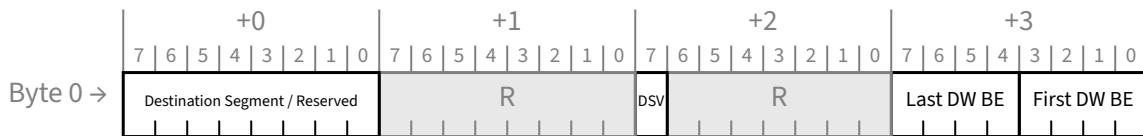


Figure 2-9 OHC-A3 §

In OHC-A3 :

- Destination Segment is Reserved if DSV is Clear.
- OHC-A3 is required as specified in § Section 2.2.5.2 .

Base 6.4 vs Base 6.3

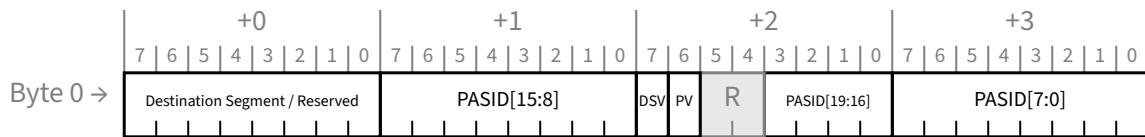


Figure 2-10 OHC-A4 §

In OHC-A4 :

- When OHC-A4 is included with a TLP, if the PASID is not known or has not been assigned, then the PV ("PASID Valid") bit must be Clear.
- The PASID field is Reserved if PV is Clear.
- The Destination Segment field is Reserved if DSV is Clear.
- OHC-A4 must be included in ID Routed Messages when Destination Segment or PASID is required.

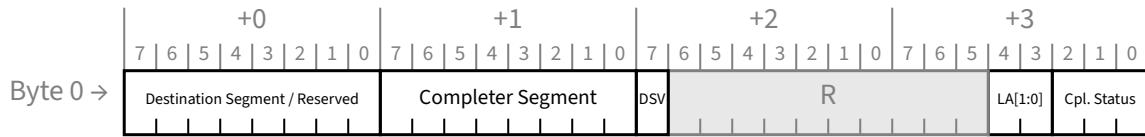


Figure 2-11 OHC-A5 §

In OHC-A5 :

- LA[1:0] is Lower Address[1:0].
- The Destination Segment field is Reserved if DSV is Clear.
- OHC-A5 is required as specified in § Section 2.2.9.2 .

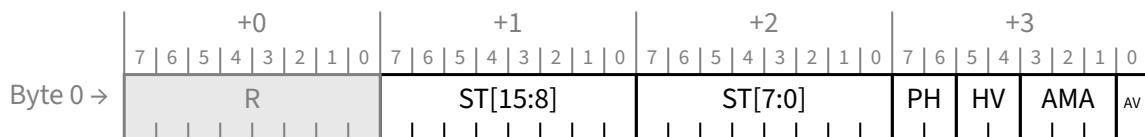


Figure 2-12 OHC-B §

In OHC-B :

- OHC-B is defined for Address Routed Requests only. When OHC-B is present on other TLPs, all OHC-B bits are Reserved.
- When TLP Processing Hints (TPH) are used OHC-B must be included with the appropriate PH and ST values.
- The PH and ST fields are qualified by the HV[1:0] ("Hints Valid") field, **↑↓ defined ↓↑ are encoded ↑** as:

Base 6.4 vs Base 6.3

- 00b: PH[1:0], ST[15:0] are not valid and are Reserved
- 01b: PH[1:0] and ST[7:0] are valid, ST[15:8] is not valid and is Reserved
- 10b: Reserved encoding, Receivers must treat as 00b.
- 11b: PH[1:0] and ST[15:0] are valid
- ↑↓AMA[2:0] ↑↓The AMA field is Reserved when AV is Clear.

```

↑↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "R", "fields": [
  {"ls15": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Requester_Segment", "addClass": "regFieldSmallText", "attr": "ro"}, {
    "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 7, "name": "PR_Sent_Counter", "addClass": "regFieldSmallText", "attr": "ro"}, {
      "lsbyte": 2, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "Stream_ID", "addClass": "regFieldSmallText", "attr": "ro"}, {
        "lsbyte": 3, "lsbit": 4, "msbyte": 3, "msbit": 6, "name": "Sub_Stream", "addClass": "regFieldVerySmallText", "attr": "ro"}, {
          "lsbyte": 3, "lsbit": 3, "msbyte": 3, "msbit": 3, "name": "RSV", "addClass": "regFieldVerySmallText", "attr": "ro"}, {
            "lsbyte": 3, "lsbit": 1, "msbyte": 3, "msbit": 1, "name": "K", "addClass": "regFieldSmallText", "attr": "ro"}, {
              "lsbyte": 0, "msbyte": 3, "msbit": 0, "name": "T", "addClass": "regFieldSmallText", "attr": "ro"}]}↓

```

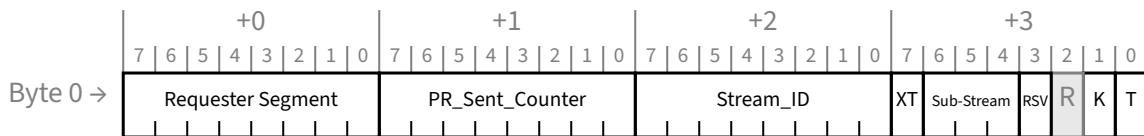


Figure 2-13 OHC-C §

For OHC-C :

- The Requester Segment field is Reserved when RSV is Clear.
- IDE TLPs must include OHC-C.
 - If Sub-Stream is 000b, 001b, or 010b, the PR_Sent_Counter, Stream_ID, K, ↑↓XT,↑ and T fields are meaningful (see § Section 6.33.5) ECN: Base 6.3 XT△
 - If Sub-Stream is ↑↓011b-110b,↑ ↑↓011b to 110b,↑ Receiver behavior is undefined.
 - For IDE Completion TLPs, the ↑↓Requester Segment field is Reserved and the RSV bit must be Clear.
- Non-IDE Request TLPs must, in some cases, also include OHC-C to indicate the Requester Segment (see Segment Rules below). When a non-IDE Completion TLP includes OHC-C, the ↑↓Requester Segment field is Reserved and the RSV bit must be ↑↓Clear,↑ ↑↓Clear,↑
 - Non-IDE TLPs with OHC-C are identified by the Sub Stream value of 111b. (see § Section 6.33.5)
 - If Sub-Stream is 111b, the PR_Sent_Counter, Stream_ID, K, ↑↓XT,↑ and T fields ECN: Base 6.3 XT△ are Reserved.
- Note: OHC-C does not include the M and P bits present in the IDE TLP Prefix . In Flit Mode, the presence of a MAC/PCRC is indicated using the TS field.

↑↓Because,↑ ↑↓Since IDE TLPs cannot be modified between the two Partner Ports, the IDE Partner Ports and the path between them must operate entirely in Non-Flit Mode or in Flit ↑↓mode,↓ ↑↓Mode,↑ Root Complexes that support peer-to-peer and Switches cannot modify IDE TLPs associated with Flow-Through Selective IDE Streams, making TLP Translation impossible. If an IDE TLP is directed out an Egress Port operating in a different mode from the Ingress Port, the IDE TLP must be dropped, and the result must be reported as a Misrouted IDE TLP error.

It is permitted to configure a Root Complex or Switch such that the Ingress Port is a terminus for an IDE connection and the Egress Port another terminus, such that the TLP is passed through the RC/Switch unprotected by IDE. Doing this requires that the RC/Switch to be trusted, and requires the Root/Switch Ports to have the ability to act as an IDE Terminus, not simply to support Flow-Through IDE.

In Flit Mode, NOP TLPs must never be transmitted as IDE TLPs. Receivers are not required to check for violations of this rule, but, if checked, Receivers must handle NOP TLPs received as IDE TLPs as Malformed TLPs.

Segment Rules:

In Flit Mode, it is possible, and in some cases required, to include Segment fields in TLPs. One benefit of the Segment fields is to enable routing Route-by-ID TLPs between Hierarchies, which are, by definition, in different Segments. Root Complexes are the only place where peer-to-peer Requests will traverse from one Hierarchy to another. Peer-to-peer Route-by-ID Message Requests can traverse Hierarchies when the Requester includes a valid Destination Segment field. Memory Requests are address routed between Hierarchies, but the associated Completions are ID routed. To aid in Root Complex routing of Completions between Hierarchies, FM Completions can include the Destination Segment field which reflects the value of the Requester Segment field from the associated NP or UIO Memory Request. This allows a Root Complex to route Non-Posted or UIO Memory Requests between Hierarchies without the need to assume ownership of each outstanding transaction for the purpose of routing the associated Completions back to the Hierarchy of the original Requester. This can lead to performance improvements for peer-to-peer transfers between Hierarchies.

A second use of the Segment fields is to improve error logging. When FM TLP headers are logged in the **AER Capability** structure the Segment fields will be included. The Segment fields improve traceability when identical Requester/Completer IDs exist in different Hierarchies. The rules in this section allow the Segment fields to be omitted in some cases to reduce FM TLP overhead. It should be noted that omitting the Segment fields in these cases could forfeit the improved error-logging traceability benefit. It is permitted to use implementation specific mechanisms to select when optional Segment fields are included (e.g., during debug) while still achieving optimal performance during normal operation by omitting non-required Segment fields.

These fields, which exist only in FM, are used to communicate Segment information:

- The Requester Segment field indicates the Hierarchy in which the Requester is located. This field exists in OHC-C and is sometimes included in Memory and Message Requests.
 - The Requester Segment Valid (RSV) bit, when Set, indicates that the Requester Segment field is valid.
 - When Requester Segment Valid (RSV) is Clear then the Requester Segment field is Reserved.
 - For TLPs with OHC-C that are not IDE TLPs, the Sub-Stream[2:0] field must be 111b, and the Stream ID, PR_Sent_Counter, **K**, XT, and T fields/bits are Reserved.
 - In earlier versions of this specification, Sub-Stream was 4 bits in Symbol 3, bits 7:4. **Bit 7 is now Reserved. If TEE IO Supported is Set, components must implement bit 7 as Reserved. If See **TEE IO Supported** is Set, components are permitted to treat bit 7 as part of Sub-Stream** for additional details.
 - IDE Requests (see § Section 6.33) other than Configuration Requests must include Requester Segment in OHC-C .
- The Completer Segment field indicates the Hierarchy in which the Completer is located. This field exists in OHC-A5 and is sometimes included in Completions.
- The Destination Segment field indicates the Hierarchy to which the TLP should be routed for ID Based Routing. In Configuration Write Requests this field is also used to configure the Segment of the completing Function. Configuration Requests in FM always include this field in OHC-A3 unless the Request had previously traversed a NFM Link. Route-by-ID Message Requests sometimes include this field in OHC-A4 . Completions sometimes include this field in OHC-A5 .

- The Destination Segment Valid (DSV) bit, when Set, indicates that the Destination Segment field is valid.
- When Destination Segment Valid (DSV) is Clear then the Destination Segment field is Reserved.

In addition to the following rules that apply specifically to Root Complexes, Requesters and Completers within Root Complexes must also follow the rules later in this section that apply to Requesters and Completers.

- All Configuration Requests transmitted by a Root Port in Flit Mode, including those initiated through the SFI Configuration Access Method, must include OHC-A3 with the DSV bit set and a valid Destination Segment. The Destination Segment is necessary for the Completer to capture its Segment as described in § Section 2.2.6.2
 - The Root Complex must indicate the correct Segment value in the Destination Segment field, even if only one Segment is implemented.
 - By definition, a Root Port and its hierarchy domain share the same Segment number.
- Completions associated with Configuration Requests must be identifiable solely by Transaction ID when received at a RP. Such Completions will not include a Destination Segment field because Configuration Requests do not include a Requester Segment field.

Errata: Base 6.3
B828^Δ◀▶

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: ROOT COMPLEX SUPPORT FOR PEER-TO-PEER NON-POSTED MEMORY TRANSACTIONS THAT TRAVERSE HIERARCHIES §

↑↓Because↓

↑↑Since↑ Segment fields aren't communicated across Links in NFM, Root Complexes take on ↑↑an↑ additional burden for peer-to-peer non-UIO NP Memory Requests that cross from one Hierarchy to another. With the loss of the Requester Segment field when a Request is translated to NFM, the Requester ID that remains in the original NP Memory Request might be indistinguishable from that of other Requesters within the hierarchy domain. Unless all Links along the path from the egress RP to the Completer are known to be in FM, Root Complexes must replace the Requester ID in peer-to-peer NP Memory Requests that cross from one Hierarchy to another. The Requester ID supplied by the Root Complex must be an ID associated with the Root Complex itself. This action is sometimes called "taking ownership" of the NP Request. It is necessary for the Root Complex to take ownership of such Requests to ensure that the Requester ID remains unique within the hierarchy domain of the egress RP, and that the associated Completions can be routed correctly by any Switches within that hierarchy domain. The egress RP also must track all such outstanding NP Memory Requests in order to route the associated Completion(s) to the Hierarchy of the original Requester within the Root Complex, as well as to restore the original Requester ID (Destination BDF/BF in FM) within the Completion(s).

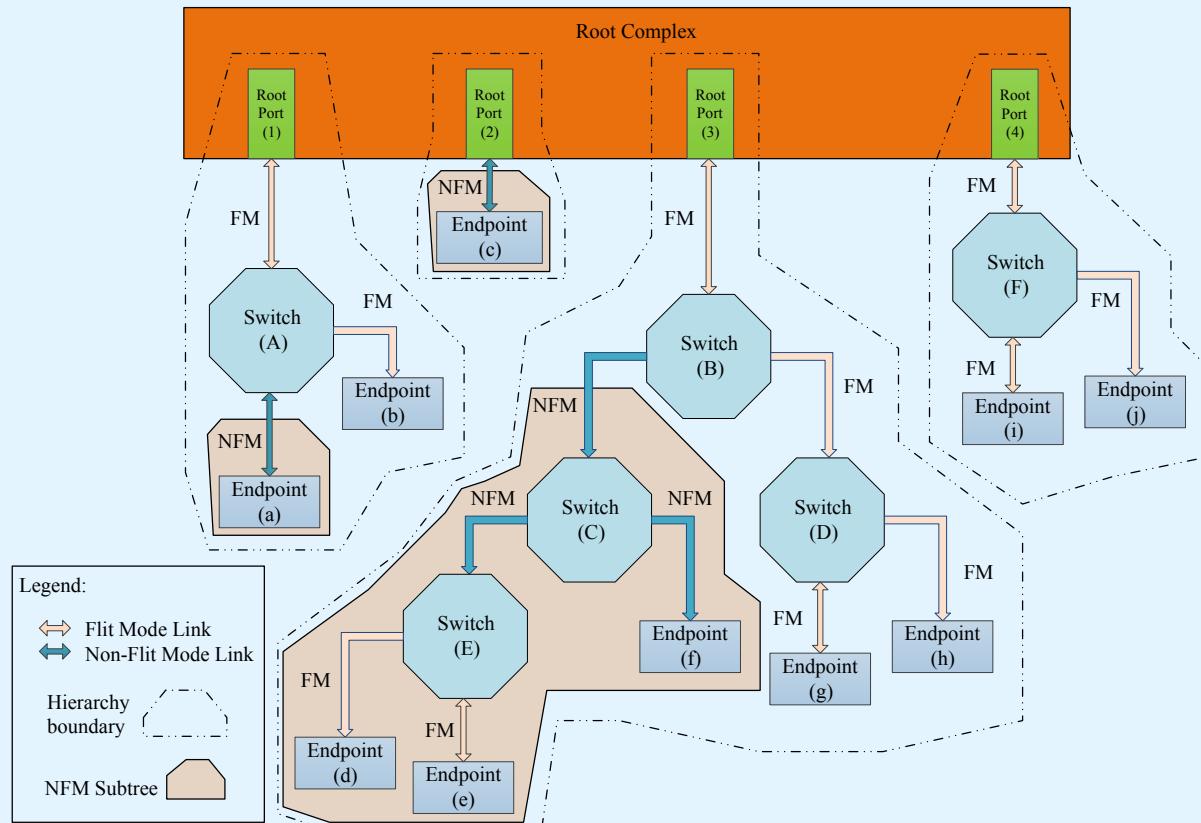


Figure 2-14 Example Topology Illustrating Multiple Segments and NFM Subtrees §

The No NFM Subtree Below This Root Port bit defaults to Clear to indicate that one or more NFM subtree(s) may exist below a Root Port. Referring to the example shown in § Figure 2-14, each Root Port is in a unique Segment/Hierarchy, and $\downarrow\downarrow$ Root Ports $\uparrow\uparrow$ 1 through $\downarrow\downarrow$ RP $\uparrow\uparrow$ 3 have NFM subtrees below the Root Port. For RP 2, the Link immediately below the $\downarrow\downarrow$ RP $\downarrow\downarrow$ $\uparrow\uparrow$ Root Port $\uparrow\uparrow$ is in NFM, but for RP 1 and $\downarrow\downarrow$ RP $\uparrow\uparrow$ 3 the Root Port cannot directly determine the existence of a NFM subtree within its hierarchy domain, and so the default value of the No NFM Subtree Below This Root Port bit ensures that the Root Complex will take ownership of NP Requests $\downarrow\downarrow$ Egressing $\downarrow\downarrow$ $\uparrow\uparrow$ Egressing $\uparrow\uparrow$ from those Root Ports. In all cases, it is necessary that system software ensure the No NFM Subtree Below This Root Port bit for a Root Port is Clear in cases where the Root Port has one or more NFM Links or subtrees below it.

$\downarrow\downarrow$ However, Root Port $\uparrow\uparrow$ 4 does not have any NFM Links below it, and therefore it is not necessary for the Root Complex to take ownership of NP Requests $\downarrow\downarrow$ Egressing $\downarrow\downarrow$ $\uparrow\uparrow$ Egressing $\uparrow\uparrow$ that Root Port. It is strongly recommended that system software Set the No NFM Subtree Below This Root Port bit in such cases, and it is strongly recommended that Root Complex implementations use the value in the No NFM Subtree Below This Root Port bit to avoid taking ownership of NP Requests when it is not necessary to do so.

Note that for non-IDE Requests directed Upstream to the RC, the existence of a NFM Link between the original Requester and the Root Port is not a factor, because the RC inherently knows the Hierarchy of the Requester based on the Ingress RP of the Request, and can add the Requester Segment if needed.

Regardless of the value of the No NFM Subtree Below This Root Port bit, a Root Complex need not apply NP Memory Request tracking mechanisms for peer-to-peer Selective IDE Stream transactions that cross from one Hierarchy to another, and IDE TLPs cannot in any case be modified between the two IDE Partner Ports. When a Selective IDE Stream is established passing peer-to-peer between Hierarchies, software must ensure that the RC supports such routing, and that the entire path between the two Partner Ports is entirely in FM.

A NFM device could be hot-added into a subtree for which the No NFM Subtree Below This Root Port bit had previously been Set. In such cases it is necessary for system software to Clear the No NFM Subtree Below This Root Port bit prior to allowing the hot-added NFM device to act as a Completer for any NP Memory Request passing peer-to-peer through the RC.

- It is not permitted to configure a Selective IDE Stream passing peer-to-peer between different Hierarchies unless it is known that the RC supports flow-through IDE between the two Root Ports, and that all Links on the path between the two Partner Ports, including both the Root Ports, are in FM.
 - Root Complexes are not required to support Selective IDE Streams passing peer-to-peer through the RC.
 - If a condition exists that precludes the RC from passing an IDE TLP associated with a Selective IDE Stream configured to flow-through the RC, then the RC must treat the TLP as a Misrouted IDE TLP error at either the Ingress Port or the Egress Port.
- If a Message or Memory Request received at a RP includes a Requester Segment that does not match the Hierarchy associated with the receiving RP, the Request must be handled by the RP as an Unsupported Request.
- A RP is permitted to add a Requester Segment indication to a non-IDE Memory Write Request, or a non-IDE Route by Address Message Request, passing peer-to-peer through the RC if that TLP did not include a Requester Segment at the ingress RP, where the Requester Segment must correspond to the Segment of the Ingress RP.
- Route by ID Message Requests received at a RP without a Destination Segment, or received in NFM, are implied to be destined for a Completer within the same Hierarchy as the Ingress RP.
- When taking ownership of an NP or UIO Memory Request passing peer-to-peer through the RC:
 - The Requester ID in the Request must be replaced with one associated with the Root Complex.

- The Request must either use the Requester Segment value associated with the hierarchy domain of the Egress RP, or must not include a Requester Segment.
- The RC is permitted to replace the Tag in the Request, and ↑↑when doing so↑ must ensure ↑↑that↑ the Transaction ID satisfies uniqueness requirements for Requests associated with the same Requester ID used for taking ownership.
 - For non-UIO Requests, the RC is permitted to change the size of the Tag. If this is done, it is permitted to use implementation specific means to determine what size of ↑↓tag↓ ↑↑Tag↑ is appropriate.
 - The Tag in the Completion(s) must be restored to the Tag from the original Request, as received at the Ingress RP, before returning those Completion(s).
- Completions associated with the Request must be identifiable solely by Transaction ID when received at the RP. Such Completions will not include a Destination Segment if the RP did not include a Requester Segment in the Request or if a NFM Link exists between the RP and the Completer.
- The Requester ID value in the Completion(s) must be restored to the Requester ID from the original Request, as received at the Ingress RP, before returning those Completion(s).
- If the RP that received the Request is in FM and OHC-A5 is returned to the Requester with the Completion(s):
 - The Destination Segment must be set to 00h and the DSV bit must be ↑↓clear↓ ↑↑Clear↑ in OHC-A5 that is returned to the original Requester.
 - The Completer Segment field must not be modified if the RP receiving the Completion is in FM and OHC-A5 was received with the Completion. The Completer Segment in OHC-A5 returned to the Requester must be set to 00h if the RP receiving the Completion is in NFM or if OHC-A5 was not received with the Completion.
- When passing an NP or UIO Memory Request peer-to-peer through the RC without taking ownership:
 - The Requester ID and Tag in the Request must not be modified.
 - For non-IDE NP Memory Requests passing peer-to-peer through the RC that do not include a Requester Segment at the Ingress RP, the RC must add a Requester Segment indication at the Egress RP, using the Segment value associated with the Ingress RP.
 - Any Completion received with the DSV bit set and a Destination Segment not matching the value associated with the hierarchy domain of the receiving RP must be routed through the RC to the specified Hierarchy.
 - The Requester ID and Tag fields returned to the Requester must not be modified from the values received with the Completion in the destination hierarchy domain.
 - If the RP that received the Request is in FM and OHC-A5 is returned to the Requester with the Completion(s) the DSV bit, Destination Segment, and Completer Segment fields must not be modified from the values received with OHC-A5 in the destination hierarchy domain.

RP Segment Exceptions – There are specific cases where a RP is not required to include Segment information:

- A RP is not required to include the Requester Segment field in any non-IDE Memory Request initiated by a Requester within the Root Complex.
- A RP is not required to include a Requester Segment field with Memory Write Requests passing peer-to-peer through the RC.
- A RP is not required to include a Requester Segment field with NP Memory Requests passing peer-to-peer through the RC if the Egress RP is taking ownership of the Request.
- A RP is not required to include the Completer Segment or Destination Segment fields in Completions associated with NP Memory Requests targeting system memory or another element of the Root Complex itself. OHC-A5 must be included if required as described in § Section 2.2.9.2.

- A RP is not required to include the Completer Segment or Destination Segment fields in Completions associated with NP Memory Requests passing peer-to-peer through the RC. OHC-A5 must be included if required as described in § Section 2.2.9.2.

Each Switch exists entirely within a single Hierarchy by definition. However, Switches are required to comprehend Segment fields in some TLP types for routing purposes. The following rules apply to Switches:

- For TLPs in FM for both the Ingress and Egress Ports, Switches must never modify, add, or remove any Segment field or the DSV/RSV bit(s) within the TLP.
- For Configuration Requests initiated in FM through the SFI Configuration Access Method on a Switch Downstream Port, the Destination Segment and DSV fields must reflect the values received in the associated Configuration Write or Read Request to the SFI CAM Data Register.
- A Switch for which ~~↓↑the↓~~ Segment Captured ~~↓↑bit↓~~ is Set must handle as a TLP Translation Egress Blocked error an NP Memory Request received at the Upstream Port destined for a Downstream Port in NFM that includes a Requester Segment that does not match the Switch's captured Segment value.
 - The Request must not be forwarded to the Downstream Port.
- If a condition exists that precludes the Switch from passing an IDE TLP associated with a Selective IDE Stream configured to flow-through the Switch without modification, then the Switch must handle the TLP as a Misrouted IDE TLP error at either the Ingress Port or the Egress Port.
- When a Switch must translate a TLP from NFM to FM:
 - If ~~↓↓Segment Captured↓~~ ~~↓↑the Segment Captured bit↓~~ is Clear, OHC-C must not be added to a Request.
 - If ~~↓↓Segment Captured↓~~ ~~↓↑the Segment Captured bit↓~~ is Set, the Switch is permitted to add OHC-C to Memory and Message Requests with the Requester Segment containing the value established when the Switch itself was configured.
 - OHC-C must not be added to Configuration Requests.
 - If any OHC-A with DSV and Destination Segment fields is added, the DSV bit must be Clear and the Destination Segment must be 00h.
 - For a Completion that requires OHC-A5 (see § Section 2.2.9.2),
 - if ~~↓↓Segment Captured↓~~ ~~↓↑the Segment Captured bit↓~~ is Set, then the Switch must ~~↓↓apply in↓~~ ~~↓↑set↓~~ the Completer Segment field ~~↓↑to↓~~ the Segment value established when the Switch itself was configured,
 - if ~~↓↓Segment Captured↓~~ ~~↓↑the Segment Captured bit↓~~ is Clear, then the Switch must ~~↓↓apply in↓~~ ~~↓↑set↓~~ the Completer Segment field ~~↓↓the value↓~~ ~~↓↑to↓~~ 00h.
- Switches must route Configuration Requests solely by the BDF fields (Destination BDF/BF in FM); the Destination Segment field must not be considered for routing.
- A Switch for which ~~↓↓Segment Captured↓~~ ~~↓↑the Segment Captured bit↓~~ is Set must route Completions and Route by ID Message Requests Upstream if DSV ~~↓↑is↓~~ Set and the Destination Segment does not match the Switch's captured Segment value.
- Completions and Route by ID Message Requests must be routed solely by Requester ID / Destination BDF / Destination Device ID if the Ingress Port is in NFM, a Destination Segment is not included (DSV bit is clear), or the included Destination Segment matches the Switch's captured Segment value.
- A Switch for which ~~↓↓Segment Captured↓~~ ~~↓↑the Segment Captured bit↓~~ is Clear must signal a TLP Translation Egress Blocked error if a Completion or Route by ID Message Request is received with DSV Set, and the TLP must not be forwarded.

- A Switch for which **↓↑Segment Captured↓↑↑the Segment Captured bit↑** is Clear must signal a TLP Translation Egress Blocked error if a received Message or Memory Request includes a Requester Segment. The Request must not be forwarded.
- A Switch for which **↓↑Segment Captured↓↑↑the Segment Captured bit↑** is Set must signal a TLP Translation Egress Blocked error if a Message or Memory Request received on a Downstream Port includes a Requester Segment that does not match the Switch's captured Segment value. The Request must not be forwarded.
- When reordering Completions with other Completions, Switches are permitted to consider Destination Segment fields included in the Completions as effectively part of the Transaction ID. When not included, the Destination Segment is implied to be the same Segment where the Completion exists.
- When reordering TLPs based on ID Based Ordering (IDO), Switches must consider Requester Segment fields included in Requests, and Destination Segment fields included in Completions, as effectively part of the Transaction ID. When the Destination Segment is not included, for reordering purposes the Destination Segment must be considered to be the same Segment where the Completion exists. When the Requester Segment is not included in a Request, Switches must assume a matching value for IDO purposes.

The following rules apply to Requesters:

- When the Requester Segment field is included in a Request it must be set to the value captured from a Configuration Write Request as described in § Section 2.2.6.2 .
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Clear all non-IDE Message and Memory Requests initiated by the Requester must not include OHC-C.
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Set all Message Requests initiated by the Requester must include OHC-C with Requester Segment.
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Set a Requester is permitted to include OHC-C with Requester Segment in Memory Requests.
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Clear, Route by ID Message Requests initiated by the Requester must not include a Destination Segment (the DSV bit must be **↓↓clear↓↑↑Clear↑**).
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Set a Requester is required to include a Destination Segment, and set the DSV bit, in Route by ID Message Requests destined for a different Hierarchy. Requesters use implementation specific means to determine the Hierarchy to which a Route by ID Message Request should be routed. When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Set the Destination Segment is required in ATS Invalidate Request, Invalidate Completion, and PRG Response Messages even if the target is in the same Hierarchy. For other Route by ID Message Requests the Destination Segment is optional when the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Set and the Requester knows, by definition or through programming, that the target of the Request is in the same Hierarchy.
- Requesters must accept any value in the Destination Segment field (if present) in received Completions.
- A Requester is not required to include the Requester Segment field in any non-IDE Memory Request.

The following rules apply to Completers:

- Completers must capture their Segment value from Configuration Write Requests as described in § Section 2.2.6.2 .
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Clear, Completers must set the Completer Segment field to 00h in any OHC-A5 that is included in a Completion.
- When the **↓↑Segment Captured↓↑↑Segment Captured↑** bit is Set, Completers must set the Completer Segment field in any OHC-A5 that is included in a Completion to the Segment value that was captured as described in § Section 2.2.6.2 .
 - If the Completion associated with the first Configuration Write Request includes OHC-A5 , the Completer Segment field must be set to the value captured from that Request.

- Completers must clear the DSV bit and set the Destination Segment field to 00h in any OHC-A5 that is included with a Completion associated with a Configuration Request.
- For an NP or UIO Memory Request received without a Requester Segment field, Completers must clear the DSV bit and set the Destination Segment field to 00h in any OHC-A5 that is included with the associated Completion(s).
- For an NP or UIO Memory Request received with a Requester Segment field, Completers must set the DSV bit and set the Destination Segment field to the value of the received Requester Segment in any OHC-A5 that is included with the associated Completion(s). See RP Segment Exceptions for cases where RPs are not required to include Segment information.
- When the ↓↑Segment Captured↓↑↑Segment Captured↑ bit is Set and an NP Memory Request is received with a Requester Segment value not matching the Completer's captured Segment value, all Completions associated with the Request must include OHC-A5.
- Completers must not qualify the acceptance of a Route by ID Message Request based on the value of the Destination Segment field in the Request.
- Completers must include OHC-A5 with a Completion if required as described in § Section 2.2.9.2 and § Section 6.33.4.
- Completers must not include OHC-A5 with a Completion when all of the following are true:
 - Completion Status is ↓↑successful↓↑↑Successful Completion↑
 - Lower Address[1:0] equal to 00b.
 - The Completer's ↓↑Segment Captured↓↑↑Segment Captured↑ bit is Clear.
- Completers are permitted to not include OHC-A5 with a Completion when all of the following are true:
 - Completion Status is ↓↑successful↓↑↑Successful Completion↑
 - Lower Address[1:0] equal to 00b.
 - The Completer's ↓↑Segment Captured↓↑↑Segment Captured↑ bit is Set.
 - The associated Request either did not include a Requester Segment field or included a Requester Segment field matching the Completer's captured Segment value.
 - The TEE-IO Supported bit is Clear or Completion is not on a Selective IDE Stream. See § Section 6.33.4.

2.2.2 TLPs with Data Payloads - Rules §

- Length is specified as an integral number of DW
- Length[9:0] is Reserved for all Messages except those that explicitly refer to a data length
 - Refer to the Message Code tables in § Section 2.2.8.
- A Function transmitting a TLP with a data payload must not allow the data payload length as indicated by the TLP's Length field to exceed the Function's applicable MPS setting. If the Function's Mixed_MPS_Supported bit is Clear or the target is host memory, the applicable MPS setting must be the Function's computed Tx_MPS_Limit, as defined below. If the Mixed_MPS_Supported bit is Set, the Function must have an implementation specific mechanism capable of supporting different MPS settings for different targets, and must handle both Request and Completion TLPs. Target-specific MPS settings are permitted to be above or below the Function's Tx_MPS_Limit, but they must never exceed the Function's Max_Payload_Size Supported field value. The Function's Tx_MPS_Limit is determined as follows:
 - For a Single-Function Device, the Tx_MPS_Limit must be its Max_Payload_Size field value, its "MPS setting".

- Base 6.4 vs Base 6.3**
- Otherwise, for an ARI Device, the Tx_MPS_Limit must be the MPS setting in Function 0. The MPS settings in other Functions of an MFD must be ignored.
 - Otherwise, for a Function in a non-ARI MFD whose MPS settings are identical across all Functions, the Tx_MPS_Limit must be the common MPS setting.
 - Otherwise, for a Function in a non-ARI MFD whose MPS settings are not identical across all Functions, the Tx_MPS_Limit must be the MPS setting in an implementation specific Function.
 - Transmitter implementations are encouraged to use the MPS setting from the Function that generated the transaction, or else the smallest MPS setting across all Functions.
 - Software should not configure the MPS setting in different Functions to different values unless software is aware of the specific implementation.
 - MPS settings apply only to TLPs with data payloads; Memory Read Requests are not restricted in length by MPS settings. The size of the Memory Read Request is controlled by the TLP's Length field.
 - The data payload size in a Received TLP as indicated by the TLP's Length field must not exceed a computed Rx_MPS_Limit for the receiving Function, as determined by MPS-related parameters as indicated below.
 - Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, the TLP must be handled as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).
 - In Flit Mode, Receivers must handle the full range of the Length field for the purpose of determining the total size of each TLP and deciding which symbol is the start of the next TLP.
 - In the receiving Function, if the Rx_MPS_Fixed bit is Set, the Rx_MPS_Limit must be the Max_Payload_Size Supported field. Otherwise, the Rx_MPS_Limit must be determined by the Max_Payload_Size field (the "MPS setting") in one or more Functions as follows:
 - For a Single-Function Device, the Rx_MPS_Limit must be its MPS setting.
 - Otherwise, for an ARI Device, the Rx_MPS_Limit must be the MPS setting in Function 0. MPS settings in other Functions must be ignored.
 - Otherwise, for an Upstream Port associated with a non-ARI MFD whose MPS settings are identical across all Functions, the Rx_MPS_Limit must be the common MPS setting.
 - Otherwise, for an Upstream Port associated with a non-ARI MFD whose MPS settings are not identical across all Functions, the Rx_MPS_Limit must be the MPS setting in an implementation specific Function.
 - Receiver implementations are encouraged to use the MPS setting from the Function targeted by the transaction, or else the largest MPS setting across all Functions.
 - Software should not configure the MPS setting in different Functions to different values unless software is aware of the specific implementation.
 - For TLPs that include data, the value in the Length field and the actual amount of data included in the TLP must match.
 - In NFM, Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, the TLP is a Malformed TLP.
 - This is a Reported Error associated with the Receiving Port (see § Section 6.2).
 - The value in the Length field applies only to data - the TLP Digest is not included in the Length.
 - When a data payload associated with a byte address is included in a TLP other than an AtomicOp Request or an AtomicOp Completion, the first byte of data following the header corresponds to the byte address closest to zero and the succeeding bytes are in increasing byte address sequence.

- Example: For a 16-byte write to location 100h, the first byte following the header would be the byte to be written to location 100h, and the second byte would be written to location 101h, and so on, with the final byte written to location 10Fh.
- The data payload in AtomicOp Requests and AtomicOp Completions must be formatted such that the first byte of data following the TLP header is the least significant byte of the first data value, and subsequent bytes of data are strictly increasing in significance. With Compare And Swap (CAS) Requests, the second data value immediately follows the first data value, and must be in the same format.
 - The endian format used by AtomicOp Completers to read and write data at the target location is implementation specific, and is permitted to be whatever the Completer determines is appropriate for the target memory (e.g., little endian, big endian, etc.). Endian format capability reporting and controls for AtomicOp Completers are outside the scope of this specification.
 - Little endian example: For a 64-bit (8-byte) Swap Request targeting location 100h with the target memory in little endian format, the first byte following the header is written to location 100h, the second byte is written to location 101h, and so on, with the final byte written to location 107h. Note that before performing the writes, the Completer first reads the target memory locations so it can return the original value in the Completion. The byte address correspondence to the data in the Completion is identical to that in the Request.
 - Big endian example: For a 64-bit (8-byte) Swap Request targeting location 100h with the target memory in big endian format, the first byte following the header is written to location 107h, the second byte is written to location 106h, and so on, with the final byte written to location 100h. Note that before performing the writes, the Completer first reads the target memory locations so it can return the original value in the Completion. The byte address correspondence to the data in the Completion is identical to that in the Request.
 - § Figure 2-15 shows little endian and big endian examples of Completer target memory access for a 64-bit (8-byte) FetchAdd. The bytes in the operands and results are numbered 0-7, with byte 0 being least significant and byte 7 being most significant. In each case, the Completer fetches the target memory operand using the appropriate endian format. Next, AtomicOp compute logic in the Completer performs the FetchAdd operation using the original target memory value and the “add” value from the FetchAdd Request. Finally, the Completer stores the FetchAdd result back to target memory using the same endian format used for the fetch.

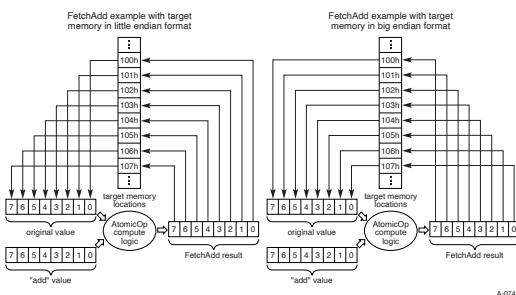


Figure 2-15 Examples of Completer Target Memory Access for FetchAdd §

IMPLEMENTATION NOTE: ENDIAN FORMAT SUPPORT BY RC ATOMICOP COMPLETERS §

One key reason for permitting an AtomicOp Completer to access target memory using an endian format of its choice is so that PCI Express devices targeting host memory with AtomicOps can interoperate with host software that uses atomic operation instructions (or instruction sequences). Some host environments have limited endian format support with atomic operations, and by supporting the “right” endian format(s), an RC AtomicOp Completer may significantly improve interoperability.

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

There is no PCI Express requirement that an RC AtomicOp Completer support the host processor's “native” format (if there is one), nor is there necessarily significant benefit to doing so. For example, some processors can use load-link/store-conditional or similar instruction sequences to do atomic operations in non-native endian formats and thus not need the RC AtomicOp Completer to support alternative endian formats.

IMPLEMENTATION NOTE: MAINTAINING ALIGNMENT IN DATA PAYLOADS §

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

2.2.3 TLP Digest Rules - Non-Flit Mode Only §

- For any TLP, a value of 1b in the TD bit indicates the presence of the TLP Digest field including an end-to-end CRC (ECRC) value at the end of the TLP.
 - A TLP where the TD bit value does not correspond with the observed size (accounting for the data payload, if present) is a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § [Section 6.2](#)).
- If an intermediate or ultimate PCI Express Receiver of the TLP does not support ECRC checking, the Receiver must ignore the TLP Digest¹³.
 - If the Receiver of the TLP supports ECRC checking, the Receiver interprets the value in the TLP Digest field as an ECRC value, according to the rules in § [Section 2.7.1](#).

¹³. An exception is an Intermediate Receiver forwarding a Multicast TLP out an Egress Port with MC_Overlay enabled. See § [Section 6.14.5](#).

2.2.4 Routing and Addressing Rules §

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

2.2.4.1 ~~Address Based~~ Address Based [Routing Rules](#) §

- Address routing is used with Memory, I/O Requests and Address Routed Messages.
- In NFM, two address formats are specified:
 - a 32-bit format with a 3 DW header (see § [Figure 2-16](#))
 - a 64-bit format with a 4 DW header (see § [Figure 2-17](#))
- In FM, five address formats are specified:
 - a 32-bit format with a 3 DW header (see § [Figure 2-18](#))
 - a 64-bit format with a 4 DW header (see § [Figure 2-19](#))
 - a 64-bit format with a 5 DW header (see § [Figure 2-20](#))
 - a 64-bit format with a 6 DW header (see § [Figure 2-21](#))
 - a 64-bit format with a 7 DW header (see § [Figure 2-22](#))

Base 6.4 vs Base 6.3

```
↑↓ [ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "addClass": "regFieldReservedText", "attr": "ro"}, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro"}, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro"}, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro"}, { "lsbyte": 7, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 through 7 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, { "lsbyte": 11, "lsbit": 2, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "attr": "ro"}, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 1, "name": "PH", "attr": "ro"} ] } ]
```

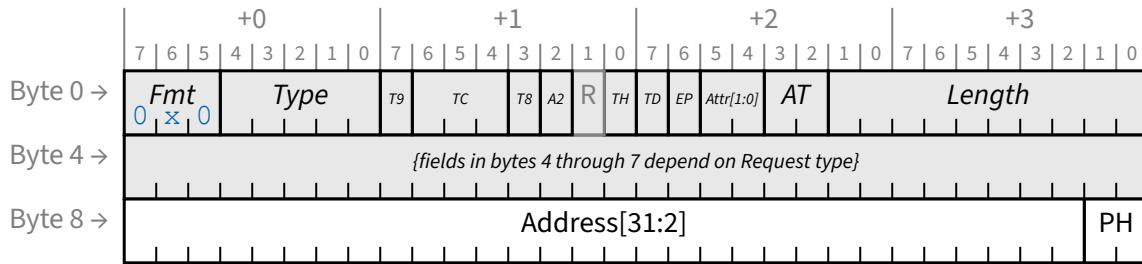


Figure 2-16 32-bit Address Routing - Non-Flit Mode

Base 6.4 vs Base 6.3

```

↓↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "1"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 2, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "attr": "ro" }, { "lsbyte": 15, "lsbit": 2, "msbyte": 12, "msbit": 7, "name": "Address[31:2]", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "PH", "attr": "ro" } ] }↓

```

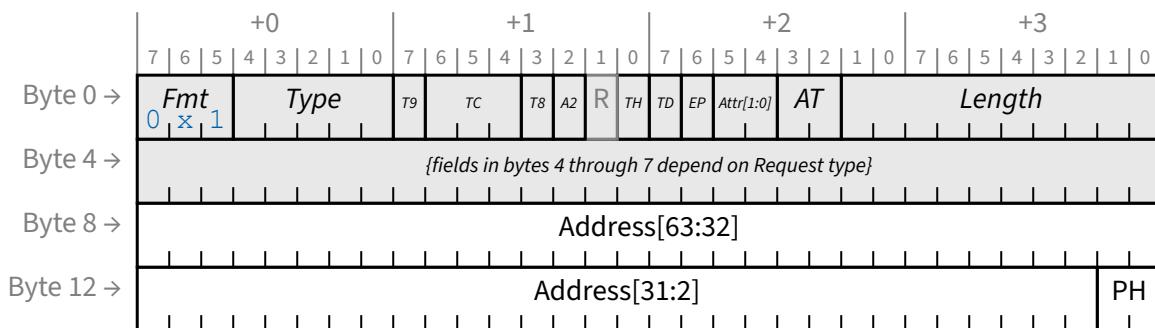


Figure 2-17 64-bit Address Routing - Non-Flit Mode §

Base 6.4 vs Base 6.3

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "attr": "ro" }, { "lsbyte": 11, "lsbit": 2, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 1, "name": "AT", "attr": "ro" } ] } }↑
```

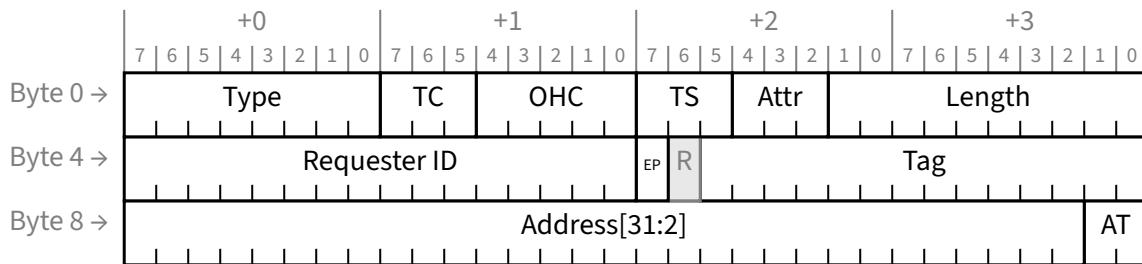


Figure 2-18 32-bit Address Routing - Flit Mode §

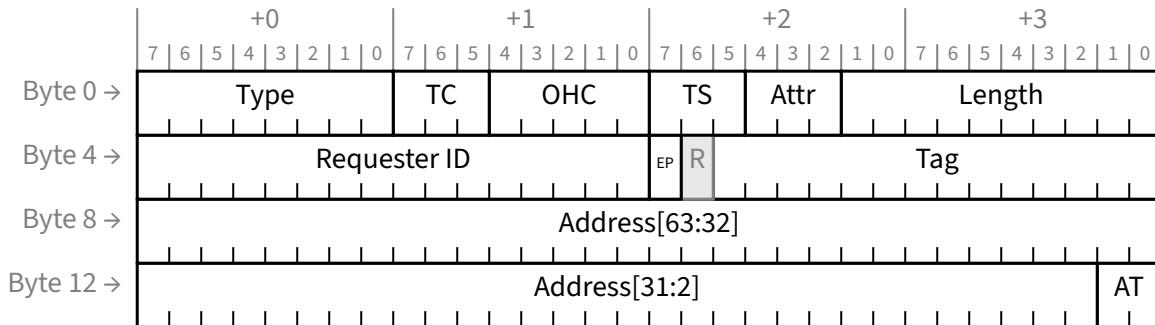


Figure 2-19 64-bit Address Routing - Flit Mode ↑- 4 DW ↑ §

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 160, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "AT", "attr": "ro" }, { "lsbyte": 19, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "{fields in bytes 16 and 19 depend on type of Request}" }, { "lsbyte": 19, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] } ↓

```

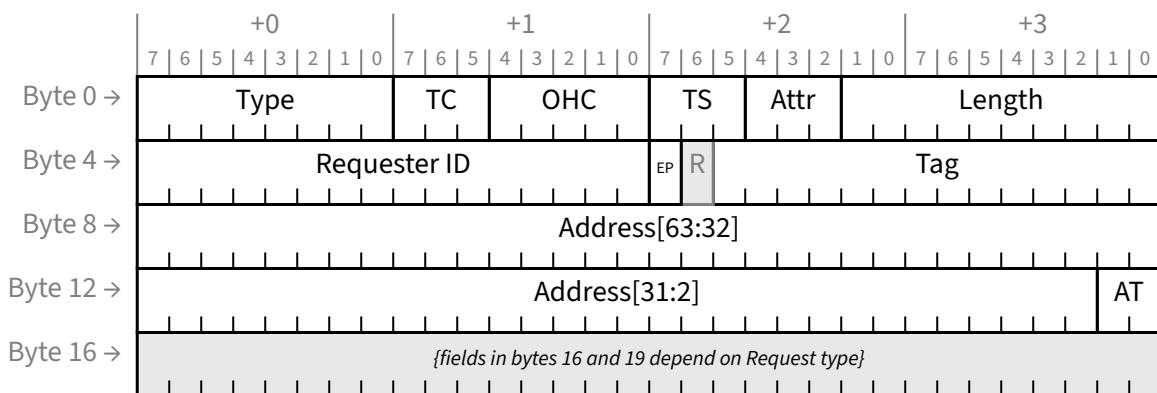


Figure 2-20 64-bit Address Routing - Flit Mode - 5 DW §

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 192, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "AT", "attr": "ro" }, { "lsbyte": 23, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "{fields in bytes 16 and 23 depend on type of Request}" }, { "lsbyte": 23, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] } ↓

```

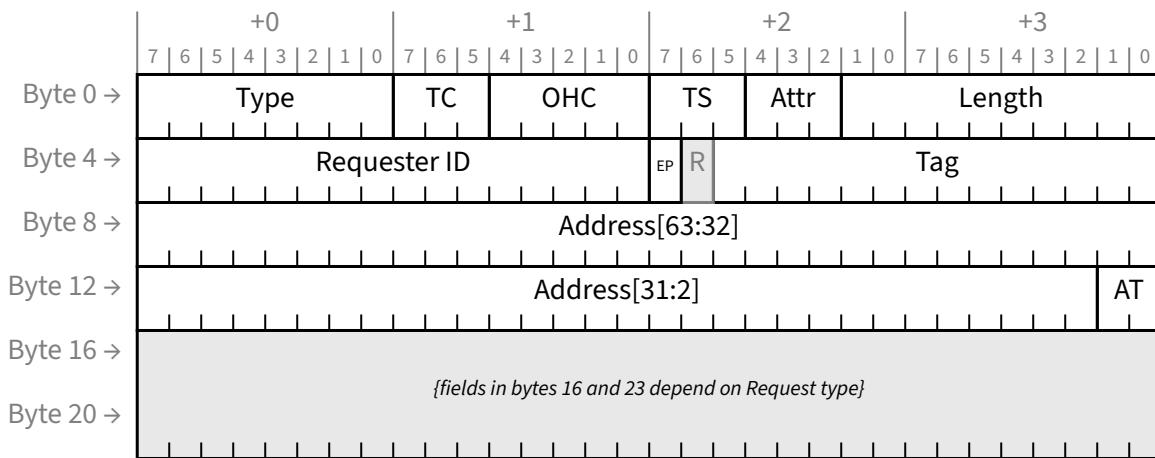


Figure 2-21 64-bit Address Routing - Flit Mode - 6 DW §

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 224, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "AT", "attr": "ro" }, { "lsbyte": 27, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "{fields in bytes 16 and 27 depend on type of Request}" }, { "lsbyte": 27, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] } ↓

```

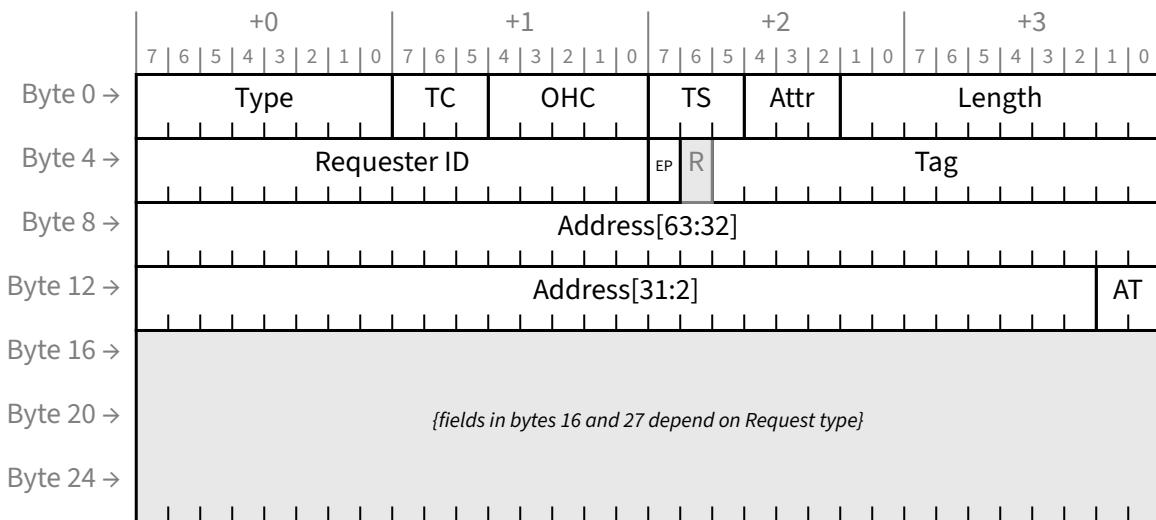


Figure 2-22 64-bit Address Routing - Flit Mode - 7 DW §

- For Memory Read, Memory Write, DMWr, and AtomicOp Requests, the Address Type (AT) field is encoded as shown in § Table 10-1 . For Address Routed Messages in Flit Mode, the Address Type (AT) field is encoded as shown in § Table 10-1 with the exception that the value of 01b is reserved. For all other Requests, the AT field is Reserved unless explicitly stated otherwise.
- If TH is Set, the PH field is encoded as shown in § Table 2-18 . If TH is Clear, the PH field is Reserved.
- Address mapping to the TLP header is shown in § Table 2-7 .

Table 2-7 Address Field Mapping §

| Address Bits | 32-bit Addressing | 64-bit Addressing |
|--------------|--------------------|---------------------|
| 63:56 | Not Applicable | Bits 7:0 of Byte 8 |
| 55:48 | Not Applicable | Bits 7:0 of Byte 9 |
| 47:40 | Not Applicable | Bits 7:0 of Byte 10 |
| 39:32 | Not Applicable | Bits 7:0 of Byte 11 |
| 31:24 | Bits 7:0 of Byte 8 | Bits 7:0 of Byte 12 |
| 23:16 | Bits 7:0 of Byte 9 | Bits 7:0 of Byte 13 |

| Address Bits | 32-bit Addressing | 64-bit Addressing |
|--------------|---------------------|---------------------|
| 15:8 | Bits 7:0 of Byte 10 | Bits 7:0 of Byte 14 |
| 7:2 | Bits 7:2 of Byte 11 | Bits 7:2 of Byte 15 |

~~↑↓ Except when explicitly required otherwise, non-UIO Memory Read, Memory Write, DMWr, and AtomicOp Requests use both formats. For Addresses below 4 GB, Requesters must use the 32-bit format. The behavior of the Receiver is not specified if a 64-bit format Request addressing below 4 GB (i.e., with the upper 32 bits of address all 0) is received. ↓~~

- The following address routed Requests must use 64-bit addressing (when addressing below 4 GB the upper 32 address bits must be to 0000 0000h):
 - All Address Routed UIO Requests
 - IDE TLPs with partial header encryption
 - This *MUST*@FLIT include Address Routed ~~↑↓ Messages. See ↓↑ Messages ↑~~¹⁴.
- ~~↑↑ Except when explicitly required otherwise, non-UIO Memory Read, Memory Write, DMWr, and AtomicOp Requests use both formats. ↑~~
 - ~~↑↑ For Addresses below 4 GB, Requesters must use the 32-bit format. The behavior of the Receiver is not specified if a 64-bit format Request addressing below 4 GB (i.e., with the upper 32 bits of address all 0) is received. ↑~~
- I/O Read Requests and I/O Write Requests use the 32-bit format.
- All agents must decode all address bits in the header - address aliasing is not allowed.

IMPLEMENTATION NOTE: PREVENTION OF ADDRESS ALIASING §

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

2.2.4.2 ID Based Routing Rules §

- ID routing is used with Configuration Requests, with ID Routed Messages, and with Completions. This specification defines several Messages that are ID Routed (see § Table F-1). Other specifications are permitted to define additional ID Routed Messages.
- ID routing uses the Bus, Device, and Function Numbers (as applicable) to specify the destination for the TLP:
 - For non-ARI Routing IDs, Bus, Device, and (3-bit) Function Number to TLP header mapping is shown in § Table 2-8.
 - For ARI Routing IDs, the Bus and (8-bit) Function Number to TLP header mapping is shown in § Table 2-9.
- In FM, Completions and ID Routed Messages with a different destination Hierarchy than the Hierarchy in which they originate must be routed to the destination Hierarchy using the Destination Segment field and then routed within the destination Hierarchy by the destination Bus, Device, and Function Numbers.

14. Earlier versions of this specification did not specify ~~↑↓ address routed message~~^{↑↑ Address Routed Message ↑} behavior when the address was below 4 GB.

- In NFM, two ID routing formats are specified, one used with a 4 DW header (see § Figure 2-23 and § Figure 2-24) and one used with a 3 DW header (see [§ Figure 2-25](#) and [§ Figure 2-26](#)).
◦ Header field locations are the same for both formats (see [§ Table 2-8](#) and [§ Table 2-9](#)).
- In FM, five ID routing formats are specified:
 - One with a 3 DW header (see [§ Figure 2-27](#))
 - One with a 4 DW header (see [§ Figure 2-28](#))
 - One with a 5 DW Header (see [§ Figure 2-29](#))
 - One with a 6 DW Header (see [§ Figure 2-30](#))
 - One with a 7 DW Header (see [§ Figure 2-31](#))

Table 2-8 Header Field Locations for non-ARI ID Routing - Non-Flit Mode

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

Table 2-9 Header Field Locations for ARI ID Routing

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

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "1"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 0, "msbit": 1, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 2, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 through 7 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[fields in bytes 12 through 15 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 3, "msbyte": 9, "msbit": 7, "name": "Device Num", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 9, "msbit": 2, "name": "Fcn Num", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Bus Number", "addClass": "regFieldVerySmallText", "attr": "ro" }
]
}

```

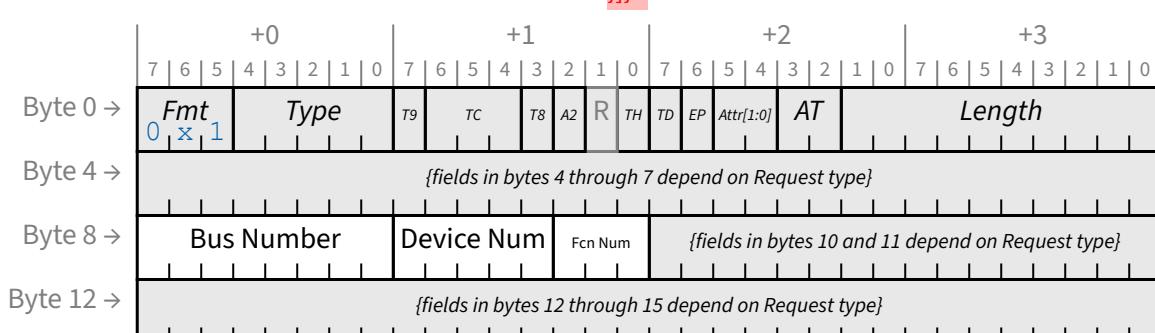


Figure 2-23 Non-ARI ID Routing with 4 DW Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "1"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 2, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[fields in bytes 12 through 15 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Bus Number", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Function Number", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Bus Number", "attr": "ro" } ] } ↓

```

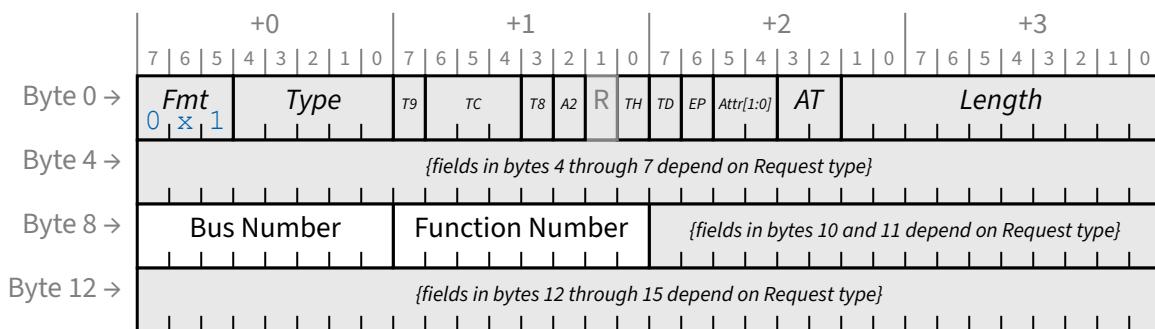


Figure 2-24 ARI ID Routing with 4 DW Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```

    ↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [
      { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 4, "msbit": 4, "name": "{fields in bytes 4 through 7 depend on type of Request}", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "{fields in bytes 10 and 11 depend on type of Request}", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 3, "msbyte": 9, "msbit": 7, "name": "Device Num", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 9, "msbit": 2, "name": "Fcn Num", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Bus Number", "addClass": "regFieldVerySmallText", "attr": "ro" }
    ]
  }

```

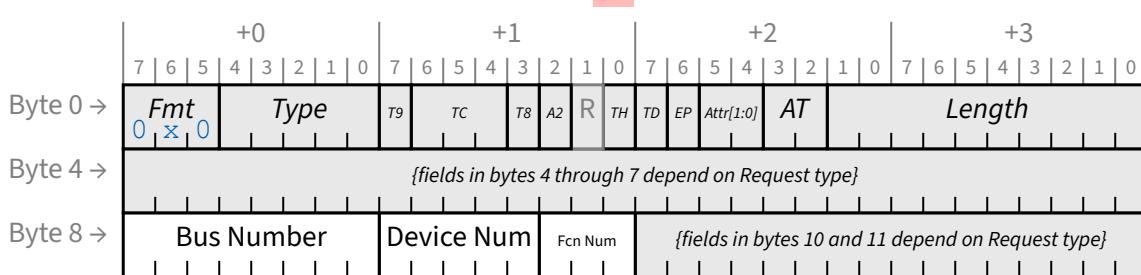


Figure 2-25 Non-ARI ID Routing with 3 DW Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```
    ], { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 through 7 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 9, "msbit": 7, "name": "Function Number", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Bus Number", "attr": "ro" } ] } ] }
```

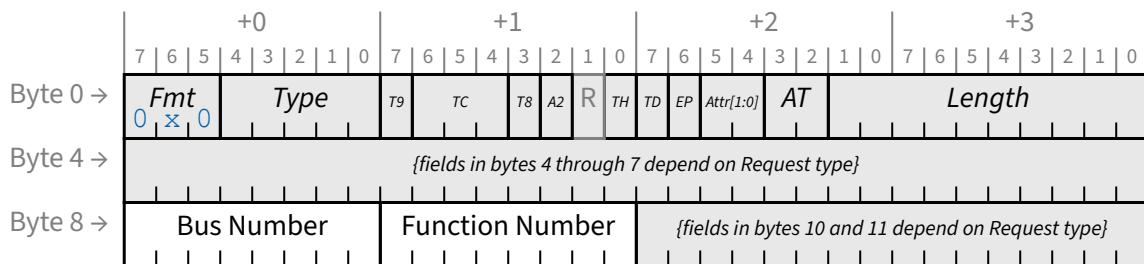


Figure 2-26 ARI ID Routing with 3 DW Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```

↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "msbit": 7, "name": "{fields in bytes 4 and 5 depend on the type of Request}", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "{fields in bytes 6 and 7 depend on the type of Request}", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "{fields in bytes 10 and 11 depend on type of Request}", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] }↓

```

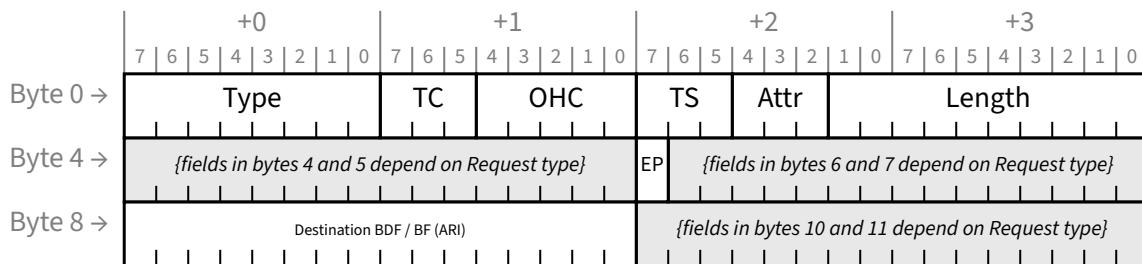


Figure 2-27 ID Routing with 3 DW Header - Flit Mode §

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "msbit": 7, "name": "[fields in bytes 4 and 5 depend on the type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "[fields in bytes 6 and 7 depend on the type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[fields in bytes 12 and 15 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] } ↓

```

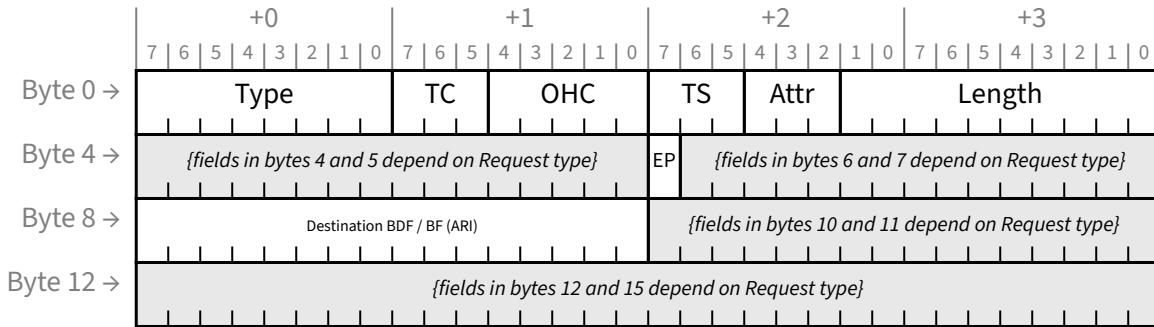


Figure 2-28 ID Routing with 4 DW Header - Flit Mode §

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 160, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "msbit": 7, "name": "[fields in bytes 4 and 5 depend on the type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "[fields in bytes 6 and 7 depend on the type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[fields in bytes 12 and 15 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] } ↓

```

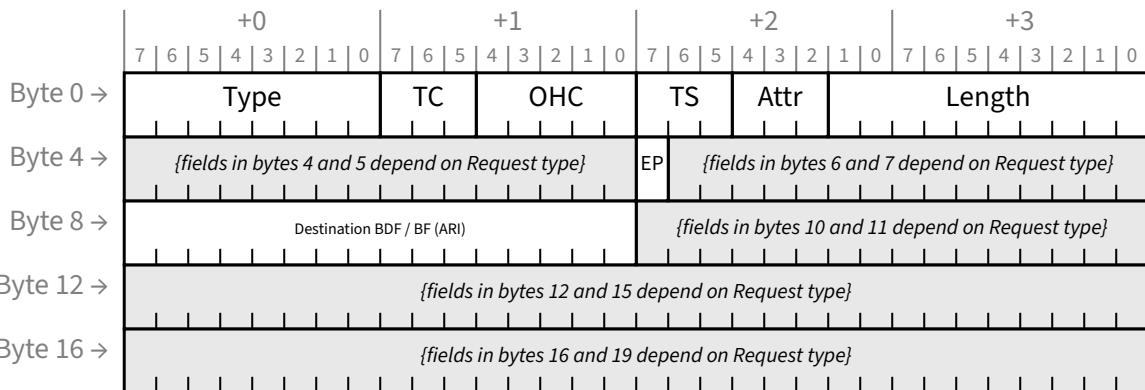


Figure 2-29 ID Routing with 5 DW Header - Flit Mode §

Base 6.4 vs Base 6.3

```

    ]]},"name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro"}, {"lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro"}, {"lsbyte": 6, "lsbit": 6, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 and 5 depend on the type of Request]", "addClass": "regFieldSmallText"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "[fields in bytes 6 and 7 depend on the type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination-BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[fields in bytes 12 and 15 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 19, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "[fields in bytes 16 and 19 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 0, "msbyte": 20, "msbit": 7, "name": "[fields in bytes 20 and 23 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}]]}↓

```

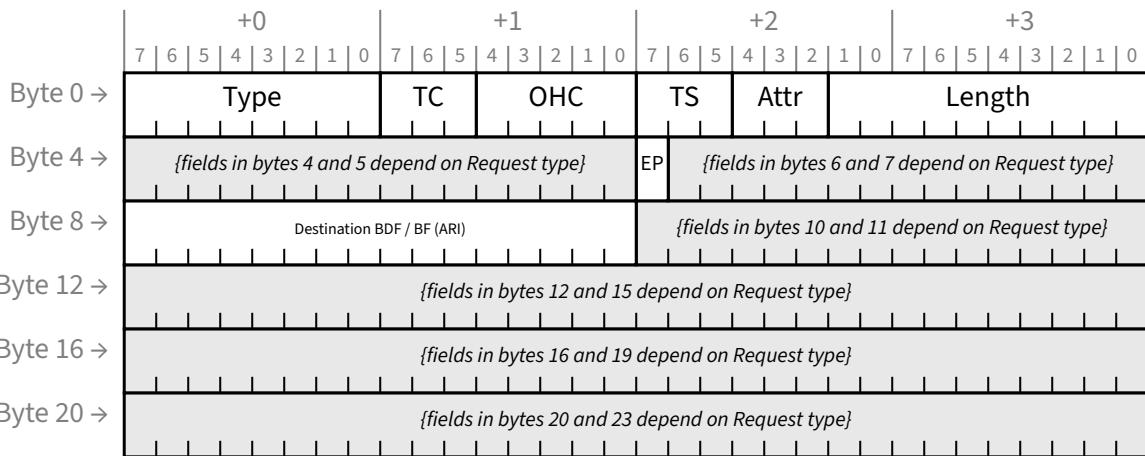


Figure 2-30 ID Routing with 6 DW Header - Flit Mode §

```

↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 224, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "Attr", "attr": "ro" },
  { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" },
  { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 and 5 depend on the type of Request]", "addClass": "regFieldSmallText" },
  { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "[fields in bytes 6 and 7 depend on the type of Request]", "addClass": "regFieldSmallText" },
  { "lsbyte": 9, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Destination BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "[fields in bytes 10 and 11 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" },
  { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[fields in bytes 12 and 15 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" },
  { "lsbyte": 19, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "[fields in bytes 16 and 19 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" },
  { "lsbyte": 23, "lsbit": 0, "msbyte": 20, "msbit": 7, "name": "[fields in bytes 20 and 23 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" },
  { "lsbyte": 27, "lsbit": 0, "msbyte": 24, "msbit": 7, "name": "[fields in bytes 24 and 27 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" } ] } ↓

```

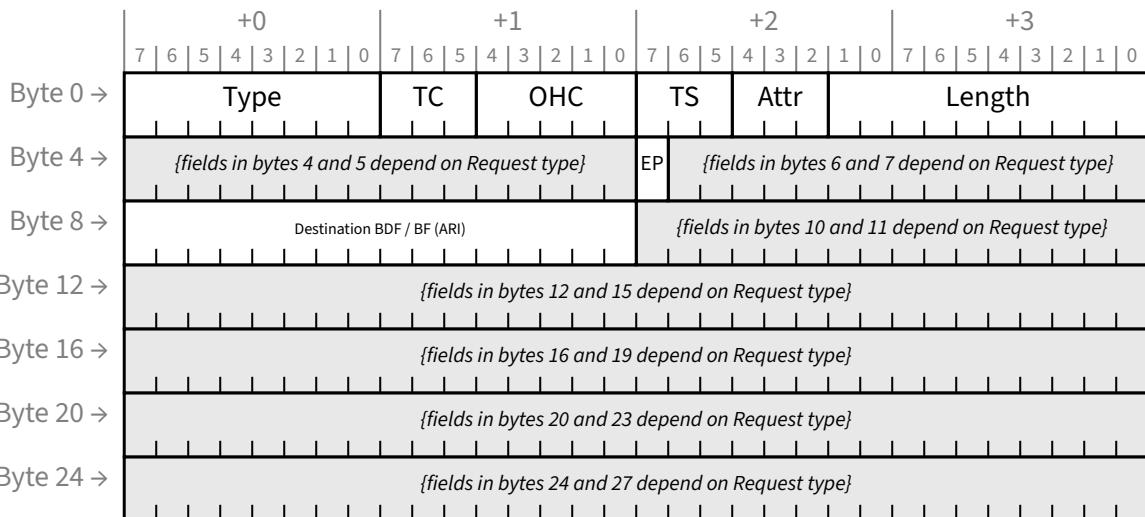


Figure 2-31 ID Routing with 7 DW Header - Flit Mode §

2.2.5 First/Last DW Byte Enables Rules §

The general function of TLP Byte Enables is similar in Non-Flit Mode and Flit Mode, however the detailed rules differ.

IMPLEMENTATION NOTE: SECURITY ISSUES ASSOCIATED WITH NON-ENABLED BYTES §

The data included with a Write or Read Completion necessarily is DW aligned, and so in cases where some bytes are not enabled, the content of the non-enabled bytes is undefined. To optimize platform security, it is strongly recommended that non-enabled bytes be filled with zeros to avoid data being inadvertently leaked (“leaky bytes”).

As a best practice, it is strongly recommended that devices receiving non-enabled bytes also ensure that the values provided in those bytes are discarded by hardware, such that the values cannot be visible to software. Hardware that fails to do so can provide a path for an attacker to observe confidential data without the need for physical access to a system.

2.2.5.1 Byte Enable Rules for Non-Flit Mode §

Byte Enables are included with Memory, I/O, and Configuration Requests. This section defines the corresponding rules. Byte Enables, when present in the Request header, are located in byte 7 of the header (see § Figure 2-32). For Memory Read Requests and DMWr Requests that have the TH bit Set, the Byte Enable fields are repurposed to carry the ST[7:0] field (refer to [§ Section 2.2.7.1.1](#) for details), and values for the Byte Enables are implied as defined below. The TH bit must only be Set in Memory Read Requests and DMWr Requests when it is acceptable to complete those Requests as if all bytes for the requested data were enabled.

- For Memory Read Requests and DMWr Requests that have the TH bit Set, the following values are implied for the Byte Enables. See § Section 2.2.7 for additional requirements.
 - If the Length field for this Request indicates a length of 1 DW, then the value for the First DW Byte Enables is implied to be 1111b and the value for the Last DW Byte Enables is implied to be 0000b.
 - If the Length field for this Request indicates a length of greater than 1 DW, then the value for the First DW Byte Enables and the Last DW Byte Enables is implied to be 1111b.

IMPLEMENTATION NOTE: READ REQUEST WITH TPH TO DWORDS WITH SIDE EFFECTS §

Memory Read Requests with the TH bit Set and that target DWORDs with side effects should only be issued when the Requester knows that completion of such reads will not create unintended side effects due to implied Byte Enable values.

```
↓↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 64, "isMessage": true, "defaultUnused": "R", "fields": [
  {"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": "[0, "x", "x"], "addClass": "regFieldReservedText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 1, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldReservedText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 through 6 depend on type of Request]", "addClass": "regFieldSmallText regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 4, "msbyte": 7, "msbit": 3, "name": "First DW BE", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 0, "name": "Last DW BE", "addClass": "regFieldVerySmallText", "attr": "ro"}]]}↓
```

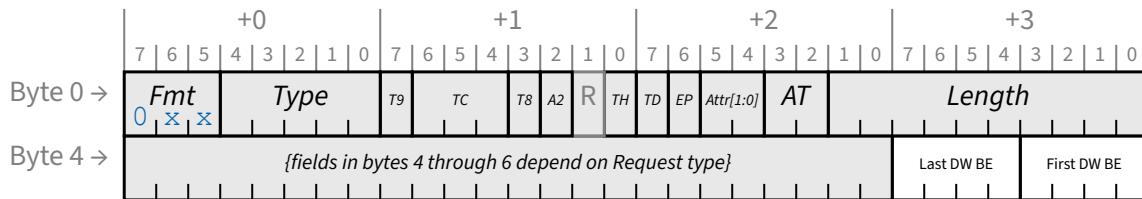


Figure 2-32 Location of Byte Enables in TLP Header - Non-Flit Mode §

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

- If the Length field for a Request indicates a length of 1 DW, this field must equal 0000b.
- If the Length field for a Request indicates a length of greater than 1 DW, this field must not equal 0000b.
- For each bit of the Byte Enables fields:
 - a value of 0b indicates that the corresponding byte of data must not be written or, if Read Side Effects exist, must not be read at the Completer.
 - a value of 1b indicates that the corresponding byte of data must be written or read at the Completer.
 - See special rules in this section regarding Memory Read Requests and DMWr Requests that have the TH bit Set.
- Non-contiguous Byte Enables (enabled bytes separated by non-enabled bytes) are permitted in the First DW BE field for all Requests with length of 1 DW.
 - Non-contiguous Byte Enable examples: 1010b, 0101b, 1001b, 1011b, 1101b
- Non-contiguous Byte Enables are permitted in both Byte Enables fields for Quad Word (QW) aligned Memory Requests with length of 2 DW (1 QW).
- All non-QW aligned Memory Requests with length of 2 DW (1 QW) and Memory Requests with length of 3 DW or more must enable only bytes that are contiguous with the data between the first and last DW of the Request.
 - Contiguous Byte Enables examples:
First DW BE: 1100b, Last DW BE: 0011b
First DW BE: 1000b, Last DW BE: 0111b
- § Table 2-10 shows the correspondence between the bits of the Byte Enables fields, their location in the Request header, and the corresponding bytes of the referenced data.

Table 2-10 Byte Enables Location and Correspondence §

| Byte Enables | Header Location | Affected Data Byte ¹⁵ |
|----------------|-----------------|----------------------------------|
| First DW BE[0] | Bit 0 of Byte 7 | Byte 0 |
| First DW BE[1] | Bit 1 of Byte 7 | Byte 1 |
| First DW BE[2] | Bit 2 of Byte 7 | Byte 2 |
| First DW BE[3] | Bit 3 of Byte 7 | Byte 3 |
| Last DW BE[0] | Bit 4 of Byte 7 | Byte N-4 |
| Last DW BE[1] | Bit 5 of Byte 7 | Byte N-3 |
| Last DW BE[2] | Bit 6 of Byte 7 | Byte N-2 |
| Last DW BE[3] | Bit 7 of Byte 7 | Byte N-1 |

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

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

IMPLEMENTATION NOTE: ZERO-LENGTH WRITE §

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

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

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

- Receiver/Completer behavior is undefined for a TLP violating the Byte Enables rules specified in this section.
- Receivers ~~may optionally~~ are permitted to check for violations of the Byte ~~Enables~~ Enable rules specified in this section. If a Receiver implementing such checks determines that a TLP violates one or more Byte ~~Enables~~ Enable rules, the TLP is a Malformed TLP. These checks are independently optional (see § Section 6.2.3.4).
 - If Byte ~~Enables~~ Enable rules are checked, a violation is a reported error associated with the Receiving Port (see § Section 6.2).
 - Byte Enable rules cannot be meaningfully checked by intermediate Receivers, and therefore must not be performed on Flow-Through IDE TLPs.

Errata: Base 6.3
B811△◀▶

IMPLEMENTATION NOTE: ZERO-LENGTH READ §

A Memory Read Request of 1 DW with no bytes enabled, or “zero-length Read,” may be used by devices as a type of flush Request. For a Requester, the flush semantic allows a device to ensure that previously issued Posted Writes have been completed at their PCI Express destination. To be effective in all cases, the address for the zero-length Read must target the same device as the Posted Writes that are being flushed. One recommended approach is using the same address as one of the Posted Writes being flushed.

The flush semantic has wide application, and all Completers must implement the functionality associated with this semantic. Since a Requester may use the flush semantic without comprehending the characteristics of the Completer, Completers must ensure that zero-length reads do not have side-effects. Note that the flush applies only to traffic in the same Traffic Class as the zero-length Read.

2.2.5.2 Byte Enable Rules for Flit Mode §

Except as defined in this section, all Byte Enable Rules in Flit Mode are the same as in Non-Flit Mode.

For all Memory Requests, it is permitted for OHC-A1 (see § Figure 2-7) to be present. OHC-A1 must be included for Requests that require any of the fields included in OHC-A1. For a Memory Request without OHC-A1 and when the Request's ~~BE~~ Byte Enable fields are not Reserved, the implied field values for a 1 DW Request are 1111b for ~~1st~~ First DW ~~BE~~ Byte Enable and 0000b for Last DW ~~BE~~ Byte Enable, and for a ~~>1 DW~~ greater than 1 DW Request is 1111b for both ~~1st~~ First DW ~~BE~~ Byte Enable and Last DW ~~BE~~ Byte Enable. If a Request requires non-Reserved BE field values other than these, then OHC-A1 must be present. When OHC-A1 is present, the PASID, PMR and ER fields are valid if and only if the PV bit is Set.

As defined in § Section 2.2.7.1 , the Byte Enable fields for AtomicOp Requests are Reserved or implied to be Reserved. If an AtomicOp Request includes OHC-A1 , its Byte Enable fields must be Reserved.

OHC-A2 must be included for all ~~↓↓IO↓↑↑I/O↑~~ Requests.

OHC-A3 must be included for all Configuration Requests.

In all cases where ~~↓↓OHC A↓↑↑OHC-Ax↑~~ is present, ~~↓↓the↓↑↑with↑~~ Byte Enable ~~↑↑fields, these↑~~ fields must be handled as defined in § Section 2.2.5.1 .

If a FM Requester uses ST[7:0] and also sets the Byte Enables to values that do not match the implied Byte Enable values specified in § Section 2.2.5.1 , the Request will not be translatable into NFM. If translation is necessary by any Routing Element between the Requester and Completer, the result will usually be a TLP Translation Egress Blocked error, subject to architected error handling rules. FM Requesters are permitted not to match the implied Byte Enable values, but are strongly recommended to consider the resulting configuration limitations.

2.2.6 Transaction Descriptor §

2.2.6.1 Overview §

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

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

§ Figure 2-33 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.

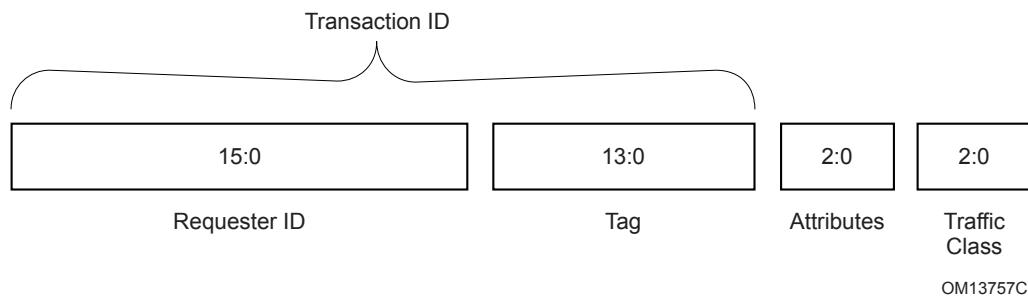


Figure 2-33 Transaction Descriptor §

2.2.6.2 Transaction Descriptor - Transaction ID Field §

The Transaction ID field consists of two major sub-fields: Requester ID and Tag as shown in § Figure 2-34 .



Figure 2-34 Transaction ID §

In some cases (defined below) the Traffic Class (TC) is also included in the Transaction ID.

The Transaction ID is used to associate Completions with Requests. There are three groups of Request/Completion types for which the Transaction ID has differing rules. The groups are distinguished by the Completion Type expected for the Request type(s) in that group. Each group forms a distinct namespace, and there is no requirement for Transaction ID uniqueness between groups. These groups and their high-level requirements are:

- Group I: Cpl / CplD, which apply to Non-UIO Requests:
 - The Transaction ID consists of Requester ID[15:0] and Tag[13:0]¹⁶
 - Requesters must assign Tag values such that Transaction ID values are unique for all outstanding Non-Posted Requests in Group I, without regard to TC or any other field.
- Group II: UIOWrCpl , which applies to UIO Memory Write (UIOMWr) Requests:
 - The Transaction ID consists of the TC[2:0], Requester ID[15:0] and Tag[13:0].
 - Requesters are permitted to assign Tag values such that multiple outstanding Requests in Group II have the same Transaction ID (see § Section 2.2.9.2)
- Group III: UIORdCplD and UIORdCpl , which apply to UIO Memory Read (UIOMRd) Requests:
 - The Transaction ID consists of the TC[2:0], Requester ID[15:0] and Tag[13:0].
 - Requesters must assign Tag values such that Transaction ID values are unique for all outstanding Requests in Group III.

Four Tag sizes are architected for ↑↑non-UIO channel↑↑ operation: 14-bit, 10-bit, ↑↓8-Bit↓↑
↑↑8-bit,↑↑ and 5-bit. ↑↑One Tag size is architected for UIO channel operation: 14-bit. § Section
2.2.6.2.1 contains rules for Tags under non-UIO channel operation. § Section 2.2.6.2.2 contains
rules for Tags under UIO channel rules.↑

Errata: Base 6.3
B807△↔

2.2.6.2.1 ↑↑Tag Rules for Non-UIO Channel Operation↑↑ §

Errata: Base 6.3
B807△↔

A given Function may support different Tag sizes when operating as a Requester versus operating as a Completer. Below are the rules regarding operational Tag sizes. Also see the “ Considerations for Implementing Larger-Tag Capabilities ” Implementation Note later in this section.

- 14-Bit Tags and 10-Bit Tags are referred to as “larger” Tags.
 - ↑↑A Request containing a 10-bit or 14 bit Tag is called **larger-Tag Request** .
 - ↑↑The 10-bit Tag Requester Enable and 14-bit Tag Requester Enable bits are called the **larger-Tag Requester Enable bits** .

Errata: Base 6.3
B807△↔

16. Not all Tag bits are used in all cases – detailed rules follow.

Base 6.4 vs Base 6.3

- **The VF 10-bit Tag Requester Enable and VF 14-bit Tag Requester Enable bits are called the VF larger-Tag Requester Enable bits .**
- 8-Bit Tags and 5-Bit Tags are referred to as “smaller” Tags.
- All Functions must support 8-Bit Tag Completer capability. **UIO Completers must support 14-bit Tags.**
- A Function that supports Flit Mode must support 14-Bit Tag Completer capability, and thus it automatically supports 10-Bit Tag Completer capability.
- Functions¹⁷ (including those in Switches) that support 16.0 GT/s **or higher** data rates **or greater** must support 10-Bit Tag Completer capability.
- A Function must not support 14-Bit Tag Requester capability unless it supports 14-Bit Tag Completer capability.
- A Function must not support 10-Bit Tag Requester capability unless it supports 10-Bit Tag Completer capability.
- In Non-Flit Mode, Tag[8] and Tag[9], are not contiguous with other Tag field bits in the TLP Header. These bits were Reserved prior to **10 Bit Tags** being architected. Requesters in Non-Flit Mode that do not support 10-Bit Tag Requester capability must set Tag[9:8] to 00b.
- RCs containing elements that indicate support for 14-Bit Tag Completer capability or 10-Bit Tag Completer capability must handle supported Tag-sized Requests correctly by all registers and memory regions supported as targets of PCIe **Requesters; e.g., Requesters (e.g., host memory targeted by DMA Requests or MMIO regions in RCiEPs.**

 - Each RP indicating support must handle such Requests received by its Ingress Port.
 - Each RCiEP indicating support must handle such Requests coming from supported internal paths, including those coming through RPs.

- If an RC contains RCiEPs that indicate support for 14-Bit Tag Requester capability or 10-Bit Tag Requester capability, the RC must handle Requests from those RCiEPs correctly by all registers and memory regions supported as targets of those **RCiEPs; e.g., host memory targeted by DMA Requests or MMIO regions in RCiEPs.**
- Receivers/Completers must handle 8-bit Tag values correctly regardless of the setting of their **Extended Tag Field Enable** bit (see § Section 7.5.3.4). Refer to **the PCI Express to PCI/PCI-X Bridge Specification** for details on the bridge handling of Extended Tags.
- Receivers/Completers that support 14-Bit Tag Completer capability or 10-Bit Tag Completer capability must handle the supported Tag-size values correctly, regardless of **the setting of** their corresponding Tag Requester Enable **bit setting. See bits (see § Section 7.5.3.16 and § Section 7.7.9.3).**
- 14-Bit Tag capability and 10-Bit Tag capability are not architected for PCI Express to PCI/PCI-X Bridges, and they must not indicate the associated Tag Requester capability or Tag Completer capability.
- If one or both **larger Tag Requester Enable bits** are Set, the following rules apply.
 - If both **larger Tag Requester Enable bits** are Set in an Endpoint¹⁸, then **14 Bit Tags** are permitted for Requests that target host memory. An implementation specific hardware mechanism in the Endpoint is permitted to limit those Requests to **10 Bit Tags** or smaller Tags, but generic software or firmware should not Set the **14 Bit Requester Enable** bit unless the host supports 14-Bit Tag Completer capability for host memory.
 - If an Endpoint¹⁹ supports sending Requests to other Endpoints (as opposed to host memory), the Endpoint must not send **larger Tag Requests** to another given Endpoint unless an implementation specific mechanism determines that the Endpoint supports the

Errata: Base 6.3
B807△

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

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

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

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

- If a PIO Requester has larger-Tag Requester capability, how the Requester determines when to use larger Tags versus smaller Tags is outside the scope of this specification. One example approach is to use smaller Tags for all PIO Requests and use larger Tags for integrated data-mover engines that use the same Requester ID. A similar approach might be used for integrated hardware that takes ownership of P2P requests.

~~↓↑For non UIO Requests/Completions, with↓↑With↓↑~~

- ~~↓↑With↓↑~~ 14-Bit Tags, determination of valid Tag values is complicated by inconsistencies in previous versions of this specification. The strongly recommended behavior is for all Tag[13:10] values except 0000b to be valid, and for 14-Bit Requesters not to generate Tag values with Tag[13:10] equal to 0000b. This enables a Requester to determine if a Completion it receives that should have a 14-Bit Tag contains an invalid Tag value. However, for backward compatibility with previous versions of this specification, 14-bit Requesters are permitted to generate any Tag[13:8] values except 00 0000b, and such Tag values are valid. ~~↓↑For UIO Requests/Completions, all Tag[13:0] values are permitted to be used.↓↑~~
- With ~~↓↑10 Bit Tags,↓↑10-Bit Tags,↑~~ all Tag[9:8] values except 00b are valid. 10-Bit Tag values with Tag[9:8] equal to 00b are invalid, and must not be generated by the Requester. This enables a Requester to determine if a Completion it receives that should have a 10-Bit Tag contains an invalid Tag value, usually caused by the Completer not supporting 10-Bit Tag Completer capability.
- If a Requester sends a ~~↓↑larger Tag Request↓↑larger-Tag Request↑~~ to a Completer that lacks the associated larger-Tag Completer capability, the returned Completion(s) will have Tags with invalid Tag values. Such Completions will be handled as Unexpected Completions²⁰, which by default are Advisory Non-Fatal Errors. The Requester must follow standard PCI Express error handling requirements.
- When a Requester handles a Completion with an invalid Tag as an Unexpected Completion, the original Request will likely incur a Completion Timeout. If the Requester handles the Completion Timeout condition in some device-specific manner that avoids data corruption, the Requester is permitted to suppress handling the Completion Timeout by standard PCI Express error handling mechanisms as required otherwise.
- If a Requester supports sending ~~↓↑larger Tag Requests↓↑larger-Tag Requests↑~~ to some Completers and ~~↓↑smaller Tag↓↑smaller-Tag↑~~ Requests to other Completers concurrently, the Requester must honor the ~~↓↑Extended Tag Field Enable↓↑Extended Tag Field Enable↑~~ bit setting for the ~~↓↑smaller Tag↓↑smaller-Tag↑~~ Requests. That is, if the bit is Clear, only the lower 5 bits of the Tag field may be non-Zero; if the bit is Set, only the lower 8 bits of the Tag field may be non-Zero.
- If a Requester supports sending ~~↓↑larger Tag Requests↓↑larger-Tag Requests↑~~ to some Completers and ~~↓↑smaller Tag↓↑smaller-Tag↑~~ Requests to other Completers concurrently, the Requester must ensure that no outstanding larger Tags can alias to an outstanding smaller Tag if any ~~↓↑larger Tag Request↓↑larger-Tag Request↑~~ is completed by a Completer that lacks larger-Tag Completer capability. See the "Using Larger Tags and Smaller Tags Concurrently" Implementation Note later in this section.
- The default value of the ~~↓↑Extended Tag Field Enable↓↑Extended Tag Field Enable↑~~ bit is implementation specific. The default value of the ~~↓↑14 Bit Tag Requester Enable↓↑14-Bit Tag Requester Enable↑~~ bit and the ~~↓↑10 Bit Tag Requester Enable↓↑10-Bit Tag Requester Enable↑~~ bit is 0b.

Errata: Base 6.3
B807△↔

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

Base 6.4 vs Base 6.3

- Receiver/Completer behavior is undefined if multiple uncompleted ~~↓↓Requests other than UIO Memory Writes,↓↑Requests,↑~~ are issued from the same Requester with non-unique Transaction ID values. In FM, Completers must be designed to handle simultaneous uncompleted Requests with non-unique Transaction ID values from Requesters that reside in different Hierarchies, as indicated by implied or explicit Segment numbers associated with each Request.
- If Phantom Function Numbers are used to extend the number of outstanding Requests, the combination of the Phantom Function Number and the Tag field must be unique for all outstanding Requests that require a Completion for that Requester, without regard to TC or any other field.
- If ~~↓↓Shadow Functions↓↑Shadow Functions↑~~ are used to extend the number of outstanding Requests, the combination of the Shadow Function Number and the Tag field must be unique for all outstanding Requests that require a Completion for that Requester, without regard to TC or any other field.
- § Table 2-11 indicates how the three tag enable bits determine the maximum tag size and permitted tag value ranges a Requester must use for different Completers and their associated paths. For a given combination of Tag enable settings, a Requester must use a Tag size within its enabled maximum and within the Tag capabilities of the Completer and its associated path. For each Request, the Requester is permitted to use a Tag size smaller than the greatest common Tag size supported by the Completer/path, but the Requester must still abide by the permitted Tag value range for the Tag size that it uses.

Errata: Base 6.3
B807 $\Delta\triangleleft$

Table 2-11 Tag Enables, Sizes, and Permitted Ranges for non-UIO Transactions §

| 14-bit Tag Requester Enable | 10-bit Tag Requester Enable | Extended Tag Field Enable | Maximum Request Tag size | Permitted range for an 8-bit Tag Completer/path | Permitted range for a 10-bit Tag Completer/path | Permitted range for a 14-bit Tag Completer/path |
|-----------------------------|-----------------------------|---------------------------|--------------------------|---|---|---|
| 0 | 0 | 0 | 5 bits | 0 to 31 | 0 to 31 | 0 to 31 |
| 0 | 0 | 1 | 8 bits | 0 to 255 | 0 to 255 | 0 to 255 |
| 0 | 1 | 0 | 10 bits | 0 to 31 | 256 to 1023 | 256 to 1023 |
| 0 | 1 | 1 | 10 bits | 0 to 255 | 256 to 1023 | 256 to 1023 |
| 1 | 0 | 0 | 14 bits | 0 to 31 | 0 to 31 | 1024 to 16383 |
| 1 | 0 | 1 | 14 bits | 0 to 255 | 0 to 255 | 1024 to 16383 |
| 1 | 1 | 0 | 14 bits | 0 to 31 | 256 to 1023 | 1024 to 16383 ²¹ |
| 1 | 1 | 1 | 14 bits | 0 to 255 | 256 to 1023 | 1024 to 16383 ²² |

Notes:

1. The permitted range for a 5-bit Tag Completer/path is always 0 to 31, so there is no column in the table to indicate this.
2. The "X-bit Tag Completer/path" is the greatest common Tag size capability of the Completer and all routing elements along the path between the Requester and the targeted Completer. If a routing element is not the targeted Completer, but detects an Uncorrectable Error with a Request, the routing element may serve as the Completer for the Request.

21. The permitted range of 1024 to 16383 is strongly recommended, but for backward compatibility with previous versions of this specification, 256 to 16383 is permitted.

22. The permitted range of 1024 to 16383 is strongly recommended, but for backward compatibility with previous versions of this specification, 256 to 16383 is permitted.

3. If a Requester supports sending ~~↑↓larger Tag Requests↓~~ ↑↓larger-Tag Requests↑ to some Completers and ~~↑↓smaller Tag↓~~ ↑↓smaller-Tag↑ Requests to other Completers concurrently, the Requester must ensure that no outstanding larger Tags can alias to an outstanding smaller Tag if any ~~↑↓larger Tag Request↓~~ ↑↓larger-Tag Request↑ is completed by a Completer that lacks larger-Tag Completer capability.
- For Posted Requests, the Tag[13:8] field is Reserved in Non-Flit Mode, and Tag[13:0] is Reserved in Flit Mode.
 - An exception to this rule is allowed for the uses defined in [MCTP-VDM].
 - In Non-Flit Mode, for Posted Requests with the TH bit Set, the Tag[7:0] field is repurposed for the ST[7:0] field (refer to § Section 2.2.7.1.1 for details). For Posted Requests with the TH bit Clear, the Tag[7:0] field is undefined and may contain any value. (Refer to § Table F-1 for exceptions to this rule for certain Vendor-Defined Messages.)
 - For Posted Requests with the TH field Clear, the value in the Tag[7:0] field must not affect Receiver processing of the Request.
 - For Posted Requests with the TH bit Set, the value in the ST[7:0] field may affect Completer processing of the Request (refer to § Section 2.2.7.1 for details).
 - A Transaction ID must be unique for each pending Transaction within a Hierarchy.
 - Transaction ID is included with all Requests and Completions.
 - The Requester ID is a 16-bit value that is unique for every PCI Express Function within a Hierarchy.
 - Functions must capture the Bus and Device Numbers²³ supplied with all Type 0 Configuration Write Requests completed by the Function and supply these numbers in the Bus and Device Number fields of the Requester ID²⁴ for all Requests initiated without the use of ~~↑↓Shadow Functions↓~~ ↑↓Shadow Functions↑ by the Device/Function. See § Section 7.9.25, for details of how the Requester ID may be modified by the use of Shadow Functions. It is recommended that Numbers are captured for successfully completed Requests only.

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

Note that the Bus Number and Device Number²⁵ may be changed at run time, and so it is necessary to re-capture this information with each and every Type 0 Configuration Write Request to the Device.

Configuration Write Requests addressed to unimplemented Functions *MUST*@FLIT not affect captured Bus and Device Numbers for implemented Functions.

- When generating Requests on their own behalf (for example, for error reporting), Switches must use the Requester ID associated with the primary side of the bridge logically associated with the Port (see § Section 7.1) causing the Request generation.
- Prior to the initial Configuration Write to a Function, the Function is not permitted to initiate Non-Posted Requests. (A valid Requester ID is required to properly route the resulting completions.)
 - Exception: Functions within a Root Complex are permitted to initiate Requests prior to software-initiated configuration for accesses to system boot device(s). Note that this rule and the exception are consistent with the existing PCI model for system initialization and configuration.
- Each Function associated with a Device must be designed to respond to a unique Function Number for Configuration Requests addressing that Device. Note: Each non-ARI Device may contain up to eight Functions. Each ARI Device may contain up to 256 Functions.

23. In ARI Devices, Functions are only required to capture the Bus Number. ARI Devices are permitted to retain the captured Bus Number on either a per-Device or a per-Function basis. If the captured Bus Number is retained on a per-Device basis, all Functions are required to update and use the common Bus Number.

24. An ARI Requester ID does not contain a Device Number field. See § Section 2.2.4.2.

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

- A Switch must forward Requests without modifying the Transaction ID, except when this is not possible due to any non-zero Tag[13:10] bits. For a Request from an Ingress Port operating in FM targeting an Egress Port operating in NFM, the presence of any non-zero Tag[13:10] bits must be handled by the Egress Port first by blocking the TLP and then reporting a TLP Translation Egress Blocked error for a Posted Request or reporting no error for a Non-Posted [↑↑Request or a UIO↑](#) Request. Such Tag bits cannot be conveyed in NFM.
- In some circumstances, a PCI Express to PCI/PCI-X Bridge is required to generate Transaction IDs for Requests it forwards from a PCI or PCI-X bus.
- In Flit Mode, Functions must capture the value of the Destination Segment supplied with all Type 0 Configuration Write Requests successfully completed by the Function. It is permitted for each Function of a Device to independently capture the Destination Segment value, or for all Functions of a Device to use the value captured by Function 0. All Functions within a Switch share a common Segment value that is captured by Functions associated with the Upstream Port. Functions also must capture the DSV bit in [Type 0 Configuration Write Requests](#) as described in the [Segment Captured](#) bit description in § [Section 7.7.9.4](#).
 - The Segment is effectively an extension of the Requester ID, but is formally defined as a distinct field to avoid confusion with the use of the term Transaction ID in Non-Flit Mode operation.
 - In systems that support multiple Segments, each Hierarchy must be associated with a single Segment. It is permitted for multiple hierarchy domains to be associated with a single Segment.
 - [↑↑The Segment Number in the Configuration Write Request must be set to the Downstream Port's Segment Number.↑](#)
- In Flit-Mode, in some circumstances, the captured Segment is also explicitly indicated in a TLP, which enables the [↑↑Transaction ID↓](#) [↑↑Completer↑](#) to [↑↑be unique between↓](#) [↑↑distinguish identical Transaction IDs in Requests coming from different↑](#) Hierarchies.

Errata: Base 6.3
B834△[↑](#)

Errata: Base 6.3
B828△[↑](#)

Errata: Base 6.3
B807△[↑](#)

IMPLEMENTATION NOTE: INCREASING THE NUMBER OF OUTSTANDING REQUESTS USING PHANTOM FUNCTIONS OR SHADOW FUNCTIONS §

To increase the maximum possible number of outstanding Requests requiring Completion beyond that possible using Tag bits alone, a device may, if the [↑↑Phantom Functions Enable↓](#) [↑↑Phantom Functions Enable↑](#) bit is Set (see § [Section 7.5.3.4](#)), or the [↑↑Shadow Functions↓](#) [↑↑Shadow Functions↑](#) Enable bit is Set (see § [Section 7.9.25.3](#)), use Function Numbers not assigned to implemented Functions to logically extend the Tag identifier. For a Single-Function Device, this can allow a significant increase in the maximum number of outstanding Requests.

When the [↑↑Phantom Functions Enable↓](#) [↑↑Phantom Functions Enable↑](#) bit is Set, unclaimed Function Numbers are referred to as Phantom Function Numbers.

Phantom Functions have a number of architectural limitations, including a lack of support by ARI Devices, Virtual Functions (VFs), and Physical Functions (PFs) when VFs are enabled. In addition, Address Translation Services (ATS) and ID-Based Ordering (IDO) do not comprehend Phantom Functions. [↑↑Shadow Functions↓](#) [↑↑Shadow Functions↑](#) have fewer limitations. Thus, for many implementations, the use of larger Tags and [↑↑Shadow Functions↓](#) [↑↑Shadow Functions↑](#) are better ways to increase the number of outstanding Non-Posted Requests.

IMPLEMENTATION NOTE: CONSIDERATIONS FOR IMPLEMENTING LARGER-TAG CAPABILITIES §

The use of "larger" (i.e., 10-bit or 14-bit) Tags enables a Requester to increase its number of outstanding Non-Posted Requests (NPRs) substantially, which for very high rates of NPRs or very large round-trip times can avoid Tag availability from becoming a bottleneck. The following formula gives the basic relationship between payload bandwidth, number of outstanding NPRs, and other factors:

$BW = S * N / RTT$, where

BW = payload bandwidth

S = transaction payload size

N = number of outstanding NPRs

RTT = transaction round-trip time

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

In configurations where a Requester with larger-Tag Requester capability needs to target multiple Completers, one needs to ensure that the Requester sends larger-Tag Requests only to Completers that have sufficient larger-Tag Completer capability. This is greatly simplified if all Completers have larger-Tag capability.

For general industry enablement of larger Tags, it is strongly recommended that all Functions²⁶ support larger-Tag Completer capability. With new implementations, Completers that don't need to operate on higher numbers of NPRs concurrently themselves can generally track larger Tags internally and return them in Completions with modest incremental investment.

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

For platforms where the RC supports larger-Tag Completer capability, it is strongly recommended for platform firmware or operating system software that configures PCIe hierarchies to Set one of the larger-Tag Requester Enable bits automatically in Endpoints with larger-Tag Requester capability. This enables the important class of larger-Tag capable adapters that send Memory Read Requests only to host memory.

For Endpoints other than RCiEPs, one can determine if the RC supports larger-Tag Completer capability for each one by checking the larger-Tag Completer Supported bits in its associated RP. RCiEPs have no associated RP, so for this reason they are not permitted to have one of their larger-Tag Requester Supported bits Set unless the RC supports sufficient larger-Tag Completer capability for them. Thus, software does not need to perform a separate check for RCiEPs.

Non-Flit Mode Switches that lack 10-bit Tag Completer capability are still able to forward NPRs and Completions carrying 10-bit Tags correctly, since the two new Tag bits are in TLP Header bits that were formerly Reserved, and Switches are required to forward Reserved TLP Header bits without modification. However, if such a Switch detects an error with an NPR carrying a 10-bit Tag, and that Switch handles the error by acting as the Completer for the NPR, the resulting Completion will have an invalid 10-bit Tag. Thus, it is strongly recommended that Non-Flit Mode Switches between any components using 10-bit Tags support 10-bit Completer capability. Note

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

that Switches supporting 16.0 GT/s **↑↓or higher↑** data rates **↑↓or greater↓** must support 10-bit Tag Completer capability.

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

IMPLEMENTATION NOTE: USING LARGER TAGS AND SMALLER TAGS CONCURRENTLY §

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

For 10-bit Tags, one implementation approach is to have the Requester partition its 8-bit Tag space into 2 regions: one that will only be used for smaller Tags (8-bit or 5-bit Tags), and one that will only be used for the lower 8 bits of 10-bit Tags. Note that this forces a tradeoff between the Tag space available for 10-bit Tags and smaller Tags.

For example, if a Requester partitions its 8-bit Tag space to use only the lowest 4 bits for smaller Tags, this supports up to 16 outstanding smaller Tags, and it reduces the 10-bit Tag space by 3×16 values, supporting $768 - 48 = 720$ outstanding 10-bit Tags. Many other partitioning options are possible, all of which reduce the total number of outstanding Requests. In general, reserving N values for smaller Tags reduces 10-bit Tag space by $3 \times N$ values, and the total for smaller Tags plus 10-bit Tags ends up being $768 - 2 \times N$.

Similar implementation approaches for **↑↓14-Bit Tags↓↑↓14-Bit Tags↑** are possible, and they are straight-forward if only 14-Bit and 8-Bit/5-Bit Tags are supported. If a Requester implementation needs to handle 14-Bit, 10-Bit, and 8-Bit/5-Bit Tag sizes concurrently, the general approach of partitioning the Requester's Tag spaces still works, but the complexity increases significantly.

2.2.6.2.2 ↑↓Tag Rules for UIO Channel Operation↑ §

Errata: Base 6.3
B807△◀▷

↑↓UIO transactions must follow the Tag namespace requirements for Group II and Group III in § Section 2.2.6.2 . Additional Tag rules under UIO channel operation are much simpler compared to non-UIO channel operation:↑

Errata: Base 6.3
B807△◀▷

- ↑↓UIO Completers must support 14-Bit Tags , i.e., returning each UIO Request's entire 14-bit Tag value in any UIO Completions resulting from the Request.↑**
- ↑↓UIO Requesters are permitted to use 14-bit Tags including all Tag[13:0] values, regardless of Tag size enable settings. See Extended Tag Field Enable , 10-Bit Tag Requester Enable , and 14-Bit Tag Requester Enable .↑**
- ↑↓UIO Completers must be designed to handle simultaneous uncompleted Requests with identical Transaction ID values coming from Requesters that reside in different Hierarchies, as indicated by implied or explicit Segment numbers associated with each Request.↑**

Errata: Base 6.3
B807△◀▷

2.2.6.3 Transaction Descriptor - Attributes Field §

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

- Ordering
- Hardware coherency management (snoop)

Attributes are hints that allow, but do not require, optimizations in the handling of traffic. The level of optimization support is dependent on the target applications of particular PCI Express peripherals and platform building blocks. In Flit Mode the Attributes Field is contiguous in the TLP Header. In Non-Flit Mode, attribute bit 2 is sometimes labeled A2 and is not adjacent to bits 1 and 0 (see [Figure 2-36](#) and [Figure 2-37](#)).

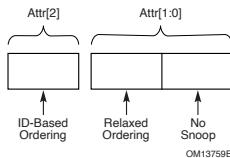


Figure 2-35 Attributes Field of Transaction Descriptor §

2.2.6.4 Relaxed Ordering and ID-Based Ordering Attributes §

Table 2-12 defines the states of the Relaxed Ordering and ID-Based Ordering attribute fields. These attributes are discussed in § Section 2.4. Note that Relaxed Ordering and ID-Based Ordering attributes are not adjacent in location (see § Figure 2-5).

Table 2-12 Ordering Attributes §

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

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

Attribute bit [2], IDO, is Reserved for Configuration Requests and I/O Requests. IDO is not Reserved for all Memory Requests, including Message Signaled Interrupts (MSI/MSI-X). IDO is not Reserved for Message Requests unless specifically prohibited. A Requester is permitted to Set IDO only if the IDO Request Enable bit in the Device Control 2 register is Set.

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

A Completer is permitted to Set IDO only if the IDO Completion Enable bit in the Device Control 2 register is Set. It is not required to copy the value of IDO from the Request into the Completion(s) for that Request. If the Completer has IDO enabled, it is recommended that the Completer set IDO for all Completions, unless there is a specific reason not to (see § Appendix E.).

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

2.2.6.5 No Snoop Attribute §

§ Table 2-13 defines the states of the No Snoop attribute field. Note that the No Snoop attribute does not alter Transaction ordering.

Table 2-13 Cache Coherency Management Attribute §

| No Snoop Attribute (b) | Cache Coherency Management Type | Coherency Model |
|------------------------|---------------------------------|--|
| 0 | Default | Hardware enforced cache coherency expected |
| 1 | No Snoop | Hardware enforced cache coherency not expected |

This attribute is not applicable and must be Clear for Configuration Requests, I/O Requests, Memory Requests that are Message Signaled Interrupts, and Message Requests (except where specifically permitted).

2.2.6.6 Transaction Descriptor - Traffic Class Field §

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

Together with the PCI Express Virtual Channel support, the TC mechanism is a fundamental element for enabling differentiated traffic servicing. Every PCI Express Transaction Layer Packet uses TC information as an Invariant label that is carried end to end within the PCI Express fabric. As the packet traverses across the fabric, this information is used at every Link and within each Switch element to make decisions with regards to proper servicing of the traffic. A key aspect of servicing is the routing of the packets based on their TC labels through corresponding Virtual Channels. § Section 2.5 covers the details of the VC mechanism.

§ Table 2-14 defines the TC encodings.

Table 2-14 Definition of TC Field Encodings §

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

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

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

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

The general requirements for Memory, I/O, and Configuration Requests [↑↑are↑](#) similar in Non-Flit Mode and Flit Mode, however some specific rules differ. Rules that are common between Non-Flit Mode and [↑↓Flit Mode↓](#) [↑↑Flit Mode↑](#) follow, with rules that are specific to each in subsequent sub-sections.

2.2.7.1 Non-Flit Mode §

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

- All Memory, I/O, and Configuration Requests include the following fields in addition to the common header fields:
 - Requester ID[15:0] and Tag[9:0], forming the Transaction ID. In Non-Flit Mode, the Tag field is 10 bits.
 - Last DW BE[3:0] and First DW BE[3:0]. For Memory Read Requests, DMWr Requests, and AtomicOp Requests with the TH bit Set, the byte location for the Last DW BE[3:0] and First DW BE [3:0] fields in the header are repurposed to carry ST[7:0] field.
 - For Memory Read Requests and DMWr Requests with the TH bit Clear, see § [Section 2.2.5](#) for First/Last DW Byte Enable Rules.
 - For AtomicOp Requests and DMWr Requests with TH bit Set, the values for the DW BE fields are implied to be Reserved. For AtomicOp Requests with TH bit Clear, the DW BE fields are Reserved.

For Memory Requests, the following rules apply:

- Memory Requests route by address, using either 64-bit or 32-bit Addressing (see [↑↑\\$ Figure 2-36↑](#) and [↑↑\\$ Figure 2-37↑](#)).
- For Memory Read Requests, Length must not exceed the value specified by Max_Read_Request_Size (see § [Section 7.5.3.4](#)).
- For AtomicOp Requests, architected operand sizes and their associated Length field values are specified in § [Table 2-15](#) . If a Completer supports AtomicOps, the following rules apply. The Completer must check the Length field value. If the value does not match an architected value, the Completer must handle the TLP as a Malformed TLP. Otherwise, if the value does not match an operand size that the Completer supports, the Completer must handle the TLP as an Unsupported Request (UR). This is a reported error associated with the Receiving Port (see § [Section 6.2](#)).

Table 2-15 Length Field Values for AtomicOp Requests §

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

- A FetchAdd Request contains one operand, the “add” value.
- A Swap Request contains one operand, the “swap” value.
- A CAS Request contains two operands. The first in the data area is the “compare” value, and the second is the “swap” value.

- For AtomicOp Requests, the Address must be naturally aligned with the operand size. The Completer must check for violations of this rule. If a TLP violates this rule, the TLP is a Malformed TLP. This is a reported error associated with the Receiving Port (see § Section 6.2).
- Requests must not specify an Address/Length combination that causes a Memory Space access to cross a 4-KB boundary.
 - Receivers may optionally check for violations of this rule. If a Receiver implementing this check determines that a TLP violates this rule, the TLP is a Malformed TLP.
 - If checked, this is a reported error associated with the Receiving Port (see § Section 6.2).
 - It is recommended that this optional check only occur in Completers and never in intermediate Receivers.
 - Intermediate Receivers are not permitted to implement this check for TLPs with Reserved Type values (see § Table 2-5). The relationship between the TLP Length field and the length of the affected memory range depends on the Request Type (for an example where they are different, see AtomicOp CAS Request).
 - For AtomicOp Requests, the mandatory Completer check for natural alignment of the Address (see above) already guarantees that the access will not cross a 4-KB boundary, so a separate 4-KB boundary check is not necessary.
 - If a 4-KB boundary check is performed for AtomicOp CAS Requests, this check must comprehend that the ~~↑↓TLP↓ ↑↑TLP's↑~~ Length ~~↑↑field↑~~ value is based on the size of two operands, whereas the access to Memory Space is based on the size of one operand.

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "1"], "attr": "ro" },
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldVerySmallText", "attr": "ro" },
  { "lsbyte": 6, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Requester ID", "attr": "ro" },
  { "lsbyte": 6, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Last DW BE", "attr": "ro" },
  { "lsbyte": 7, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Tag", "attr": "ro" },
  { "lsbyte": 7, "lsbit": 4, "msbyte": 7, "msbit": 7, "name": "First DW BE", "attr": "ro" },
  { "lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[32:3]", "attr": "ro" },
  { "lsbyte": 15, "lsbit": 15, "msbyte": 1, "msbit": 1, "name": "PH", "attr": "ro" }
]}]
```

Base 6.4 VS Base 6.3

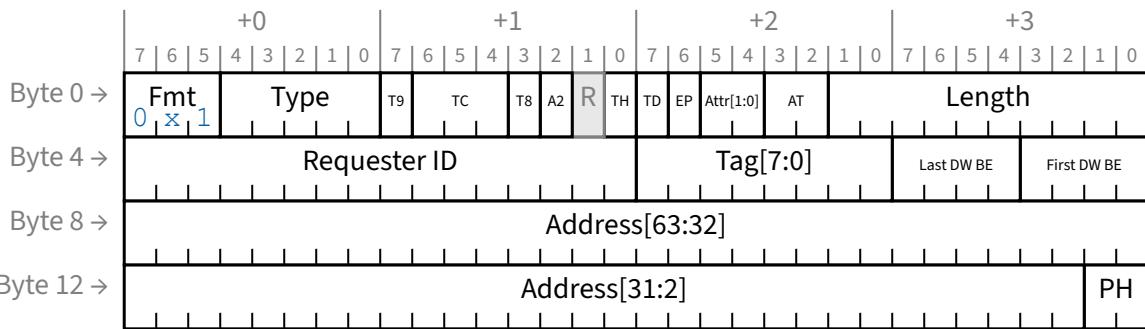


Figure 2-36 [Memory Request Header Format](#) for 64-bit Addressing [of Memory](#) [Header - Non-Flit Mode](#)

```

    "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [
      {"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Tag", "attr": "ro"}, {"lsbyte": 7, "lsbit": 4, "msbyte": 7, "msbit": 7, "name": "Last DW BE", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 3, "name": "First DW BE", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 2, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 1, "name": "PH", "attr": "ro"}]
  
```

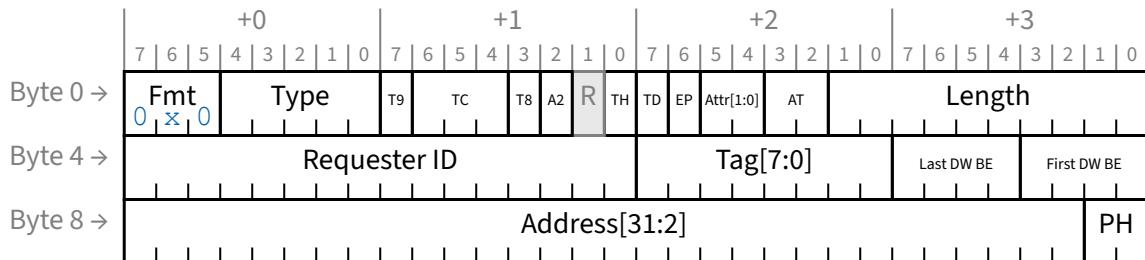


Figure 2-37 [Memory Request Header Format](#) for 32-bit Addressing [of Memory](#) [Header - Non-Flit Mode](#)

IMPLEMENTATION NOTE: GENERATION OF 64-BIT ADDRESSES §

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

For I/O Requests, the following rules apply:

- I/O Requests route by address, using 32-bit Addressing (see [Figure 2-38](#))
- I/O Requests have the following restrictions:
 - TC[2:0] must be 000b
 - TH is not applicable to I/O Request and the bit is Reserved
 - Attr[2] is Reserved
 - Attr[1:0] must be 00b
 - AT[1:0] must be 00b. Receivers are not required or encouraged to check this.
 - Length[9:0] must be 00 0000 0001b
 - Last DW BE[3:0] must be 0000b

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

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

```
↑↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [{"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldSmallText", "value": ["0", "0", "0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "value": ["R"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldSmallText", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "1"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Requester ID", "value": ["0", "0", "0", "0", "0", "0", "0", "1"], "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Tag", "attr": "ro"}, {"lsbyte": 7, "lsbit": 4, "msbyte": 7, "msbit": 7, "name": "Last DW BE", "value": ["0", "0", "0", "0"], "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 3, "name": "First DW BE", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 2, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "attr": "ro"}]}↓
```

Base 6.4 vs Base 6.3

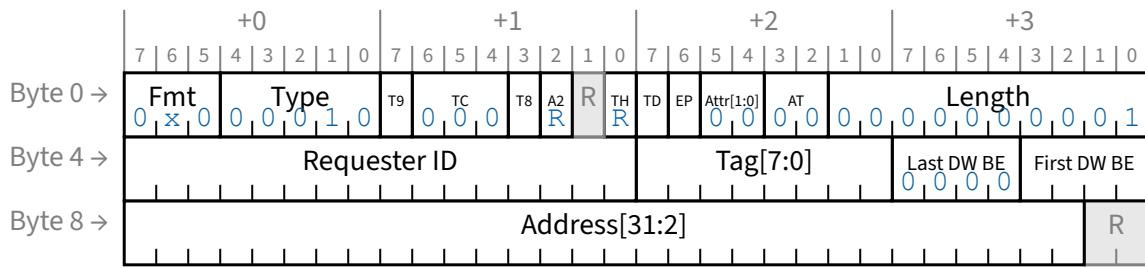


Figure 2-38 **I/O Request Header Format for I/O Transactions - Non-Flit Mode**

For Configuration Requests, the following rules apply:

- Configuration Requests route by ID, and use a 3 DW header.
- In addition to the header fields included in all Memory, I/O, and Configuration Requests and the ID routing fields, Configuration Requests contain the following additional fields (see [Figure 2-39](#)).
 - Register Number[5:0]
 - Extended Register Number[3:0]
- Configuration Requests have the following restrictions:
 - TC[2:0] must be 000b
 - TH is not applicable to Configuration Requests and the bit is Reserved
 - Attr[2] is Reserved
 - Attr[1:0] must be 00b
 - AT[1:0] must be 00b. Receivers are not required or encouraged to check this.
 - Length[9:0] must be 00 0000 0001b
 - Last DW BE[3:0] must be 0000b

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

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

```
|| { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText", "value": ["0", "0", "0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText", "value": ["R"], "isUnused": true, "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "value": ["0", "0"], "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 3, "msbit": 1, "name": "AT", "addClass": "regFieldSmallText", "value": ["0", "0"], "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "1"], "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Last DW BE", "attr": "ro" } ] }
```

```
BE", "addClass": "regFieldVerySmallText", "value": ["0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 3, "name": "First DW BE", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination ID", "attr": "ro"}, {"lsbyte": 10, "lsbit": 4, "msbyte": 10, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro"}, {"lsbyte": 10, "lsbit": 0, "msbyte": 10, "msbit": 3, "name": "Ext Reg Num", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 2, "msbyte": 11, "msbit": 7, "name": "Register Number", "addClass": "regFieldVerySmallText", "attr": "ro"}]} ] } ] }
```

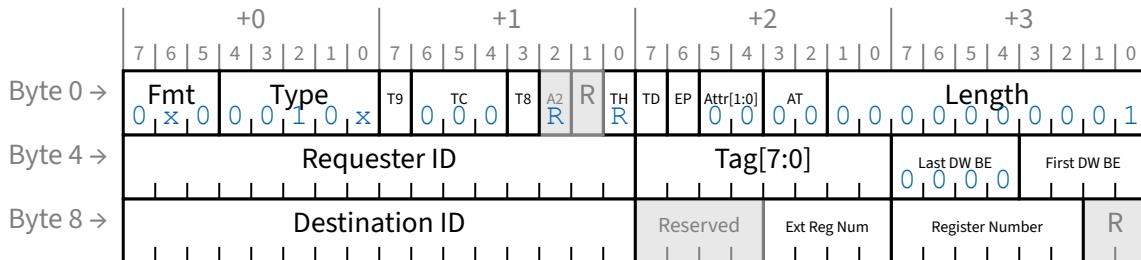


Figure 2-39 [Configuration Request Header](#) [Format for Configuration Transactions](#) - Non-Flit Mode §

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

2.2.7.1.1 TPH Rules [Non-Flit Mode](#) §

- Two formats are specified for TPH. The [Baseline](#) [TPH format without the TPH TLP Prefix](#) (see § Figure 2-41 [and](#) § Figure 2-42 [, § Figure 2-43 , and § Figure 2-44](#)) must be used for all Requests that provide TPH. The [Extended TPH](#) format with the [optional](#) TPH TLP Prefix extends the TPH fields (see § Figure 2-40) to provide additional bits for the Steering Tag (ST) field.

```
{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "Reserved", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "[see Section 2.2.1]", "addClass": "regFieldSmallText", "attr": "ro"}, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 7, "name": "ST [15:8]", "addClass": "regFieldSmallText", "attr": "ro"}, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "AMA [2:0]", "addClass": "regFieldVerySmallText", "attr": "ro"}, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 4, "name": "AV", "addClass": "regFieldVerySmallText", "attr": "ro"} ] }
```

[Check and fix Section 2.2.1 reference in figure](#)

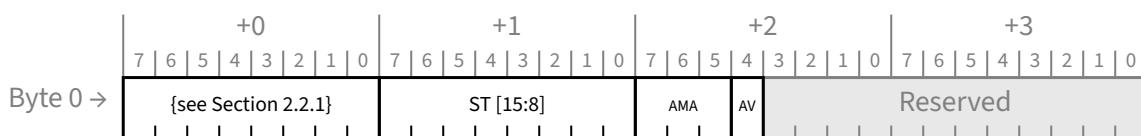


Figure 2-40 TPH TLP Prefix [Non-Flit Mode](#) §

- The optional TPH TLP Prefix is used to provide additional TPH information.
 - The presence of a TPH TLP Prefix is determined by decoding byte 0.

Table 2-16 TPH TLP Prefix Bit Mapping

| Fields | TPH TLP Prefix |
|--|--------------------|
| ST[15:8] | Bits 7:0 of byte 1 |
| ↑↓AMA[2:0]↓ ↑↑AMA↑ | Bits 7:5 of byte 2 |
| AV | Bit 4 of byte 2 |
| Reserved | Bits 3:0 of byte 2 |
| Reserved | Bits 7:0 of byte 3 |

- The TPH TLP Prefix is used to send a non-Zero value for any of:
 - AMA
 - ST[15:8]
- For Requests that target Memory Space, a value of 1b in the TH bit indicates the presence of TPH in the TLP header and optional TPH TLP Prefix (if present).
 - The TH bit must be Set for Requests that provide TPH.
 - The TH bit is permitted to be Set for Requests with a TPH TLP Prefix. When the TH bit is 1b, then ST[15:8] is present and meaningful in the TPH TLP Prefix.
 - When the TH bit is Clear, the PH field is Reserved.
 - The TH bit and the PH field are not applicable and are Reserved for all other Requests.
- For Requests that target Memory Space, the TPH TLP Prefix may be present if the value of the TH bit is 0b. When the ~~↑↓AMA Valid (AV)↓~~ ~~↑↑AV↑~~ bit is 1b and the ~~↑↓TPH TLP Prefix↓~~ ~~↑↑TPH TLP Prefix↑~~ is present, AMA is present and meaningful in the ~~↑↓TPH TLP Prefix.↓~~ ~~↑↑TPH TLP Prefix.↑~~
- For Requests that target Memory Space with the AT field not set to 10b, the AMA field in the TPH TLP Prefix is Reserved.
- The Processing Hints (PH) fields mapping is shown in § Figure 2-41, § Figure 2-42 and § Table 2-17.

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "1"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 2, "name": "AT", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 0, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[fields in bytes 4 through 7 depend on type of Request]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 15, "lsbit": 2, "msbyte": 12, "msbit": 7, "name": "Address[31:2]", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "PH", "attr": "ro" } ] }↓

```

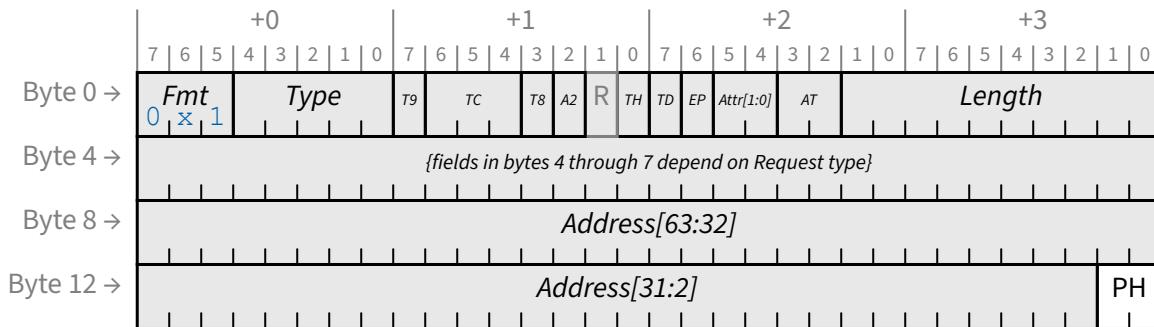


Figure 2-41 Location of PH[1:0] in a 4 DW Request Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```

    "l↓v{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [
      { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "TC", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 2, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 11, "msbyte": 11, "msbit": 1, "name": "PH", "attr": "ro" } ] } ↓
  
```

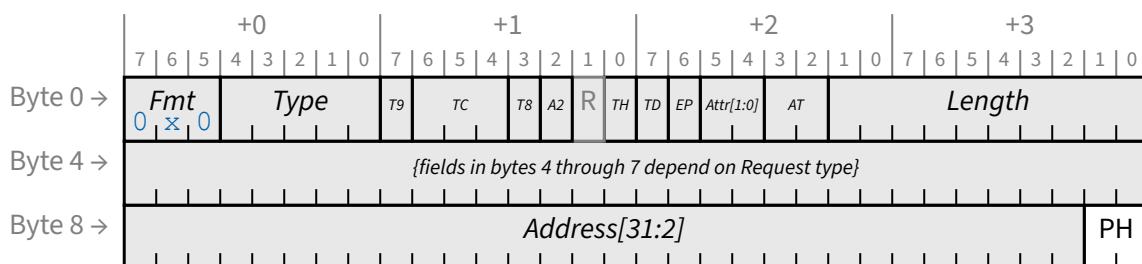


Figure 2-42 Location of PH[1:0] in a 3 DW Request Header - Non-Flit Mode §

Table 2-17 Location of PH[1:0] in TLP Header ↑↑- Non-Flit Mode ↑ §

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

- The PH[1:0] field provides information about the data access patterns and is defined as described in § Table 2-18.

Table 2-18 Processing Hint Encoding §

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

Base 6.4 vs Base 6.3

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

The Steering Tag (ST) fields are mapped to the TLP header as shown in § Figure 2-43 , § Figure 2-44 and [§ Table 2-19](#)

```

↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 64, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "addClass": "regFieldReservedText", "attr": "ro" }, {
        "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, {
            "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText",
            "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText
            regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                    "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                        "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                            "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                                "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                                    "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                                        "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, {
                                            "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "addClass": "regFieldReservedText", "attr": "ro" }, {
                                                "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "ST[7:0]", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                                                    "lsbyte": 7, "lsbit": 4, "msbyte": 3, "msbit": 0, "name": "Last DW BE", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, {
                                                        "lsbyte": 7, "lsbit": 3, "msbyte": 7, "msbit": 0, "name": "First DW BE", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" } ] }↓

```

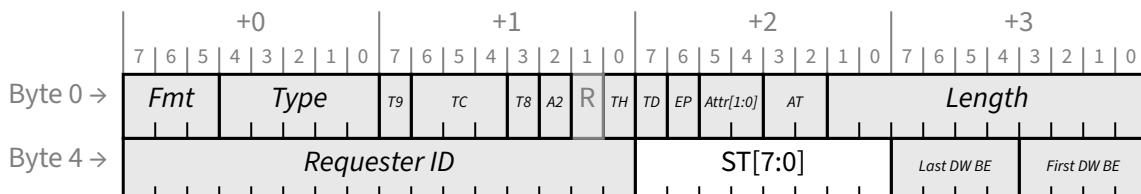


Figure 2-43 Location of ST[7:0] in the Memory Write Request Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```

↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 64, "isMessage": true, "defaultUnused": "R", "fields": [
    { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "addClass": "regFieldReservedText", "attr": "ro" }, {
    "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "addClass": "regFieldReservedText", "attr": "ro" }, {
    "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldVerySmallText regFieldReservedText",
    "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldVerySmallText
    regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass":
    "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name":
    "A2", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1,
    "msbit": 0, "name": "TH", "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2,
    "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldVerySmallText regFieldReservedText",
    "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText
    regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass":
    "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT",
    "addClass": "regFieldVerySmallText regFieldReservedText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2,
    "msbit": 1, "name": "Length", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4,
    "msbit": 7, "name": "Requester ID", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0,
    "msbyte": 6, "msbit": 7, "name": "Tag", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 7,
    "msbyte": 7, "msbit": 7, "name": "ST[7:0]", "addClass": "regFieldReservedText", "attr": "ro" } ] }↓

```

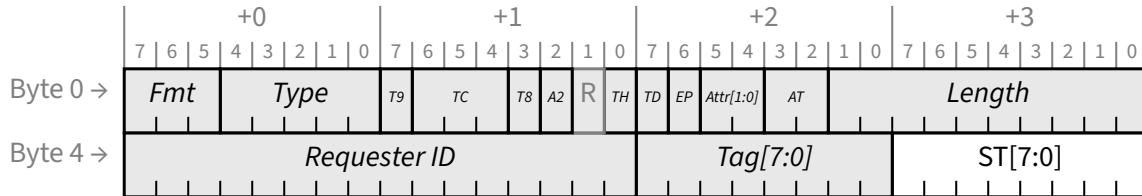


Figure 2-44 Location of ST[7:0] in Memory Read, DMWr, and AtomicOp Request Headers - Non-Flit Mode §

Table 2-19 Location of ST[7:0] in TLP Headers ↑↑- Non-Flit Mode ↑ §

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

- ST[7:0] field carries the Steering Tag value
 - A value of Zero indicates no Steering Tag preference
 - A total of 255 unique Steering Tag values are provided
- A Function that does not support the TPH Completer or Routing capability and receives a transaction with the TH bit Set is required to ignore the TH bit and handle the Request in the same way as Requests of the same transaction type without the TH bit Set.

2.2.7.2 Flit Mode §

Except as stated, rules that apply in Non-Flit Mode also apply in Flit Mode.

- All Memory, I/O, and Configuration Requests include the following fields in addition to the common header fields:
 - A Transaction ID, consisting of Requester ID[15:0] and Tag[13:0], and, for Memory Requests, sometimes also including the Requester Segment[7:0]

Base 6.4 vs Base 6.3

- Byte Enable rules for Flit Mode are covered in § Section 2.2.5.2 . There are several notable differences from the Byte Enable rules for Non-Flit Mode covered in § Section 2.2.5.1 .
- For non-UIO Memory Requests, including AtomicOp and DMWr, the rules for the formation and processing of Header Fields are the same as in Non-Flit Mode.
- For UIO Requests, the rules for the formation and processing of Header Fields are the same as in Non-Flit Mode with the following ~~↑↓exception:~~ ↑↑exceptions:
 - ~~↑↓Attr[2:1], corresponding to IDO and RO in non-UIO Memory Requests,~~ ↑↑Attr[2:1] are Reserved
 - AT[1:0] value of 01b is Reserved (See § Section 10.2.2)
 - When multiple outstanding Group II UIO Requests are issued using the same Transaction ID (see § Section 2.2.6.2), ~~↑↓all outstanding~~ ↑↑such Requests ~~↑↓using a given Transaction ID~~ must have the same value for Attr[0] (i.e., No Snoop).
- For ~~↑↓IO~~ ↑↑I/O Requests, the rules for the formation and processing of Header Fields are the same as in Non-Flit Mode.
- Configuration Requests must include OHC-A3 .
- Configuration Requests must only include ~~↑↓OHC-C~~ ↑↑OHC-C when they are associated with an IDE stream.
- UIO Requests are only defined for Flit Mode.
- ↑↑The TH bit, present in Non-Flit Mode Requests, is not supported in Flit Mode.
- The following figures illustrate currently defined Flit Mode Request Headers:
 - Reserved Requests (as indicated in § Table 2-5), are defined in § Section 2.2.4.1 and § Section 2.2.4.2 .

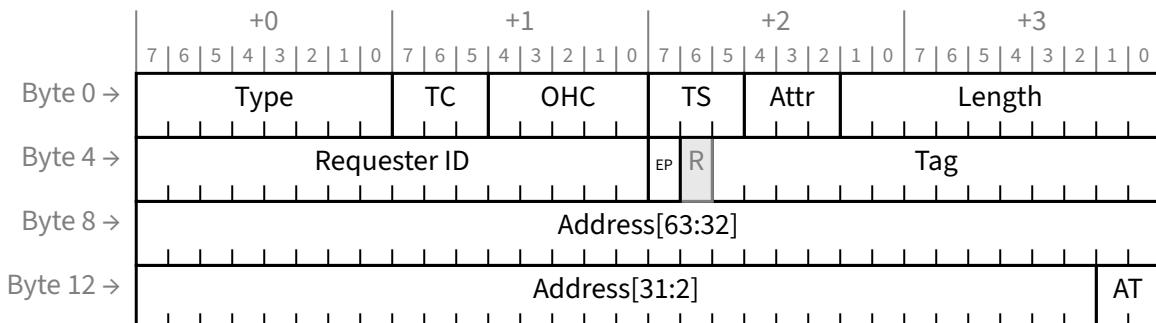


Figure 2-45 ↑↑Memory Request with 64-bit Addressing Header → Flit Mode ~~↑↓Mem64 Request~~ §

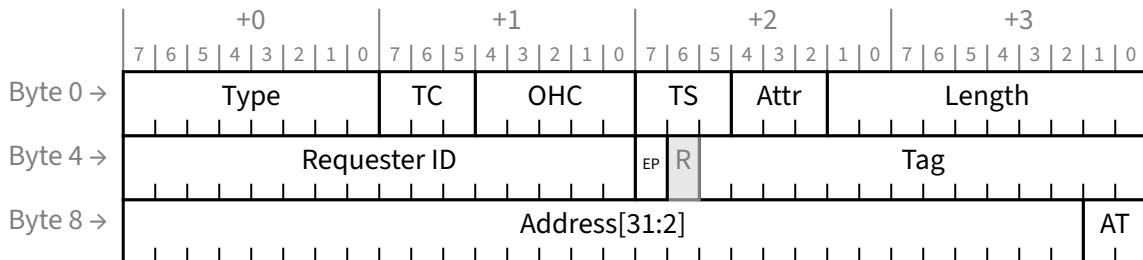


Figure 2-46 ↑↑Memory Request with 32-bit Addressing Header → Flit Mode ~~↑↓Mem32 Request~~ §

Base 6.4 vs Base 6.3

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "value": ["0", "0", "0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "1"], "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "attr": "ro" }, { "lsbyte": 11, "lsbit": 2, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 1, "name": "R", "attr": "ro" } ] }↓
```

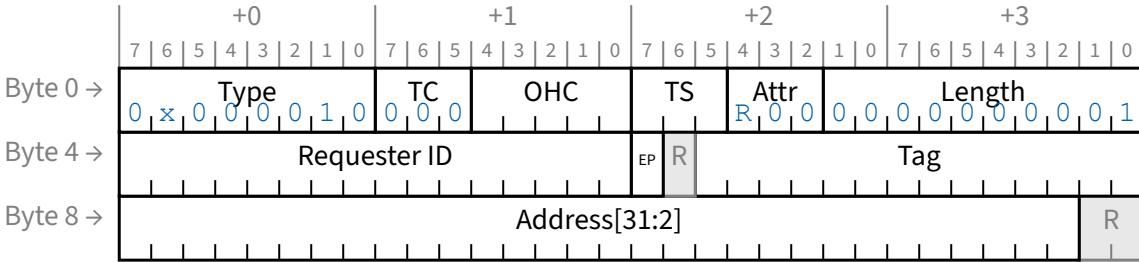


Figure 2-47 ↑↓I/O Request Header -↑ Flit Mode ↑↓I/O Request↓ §

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "value": ["0", "0", "0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "1"], "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 2, "msbyte": 10, "msbit": 3, "name": "Register Number", "addClass": "regFieldVerySmallText", "attr": "ro" } ] }↓
```

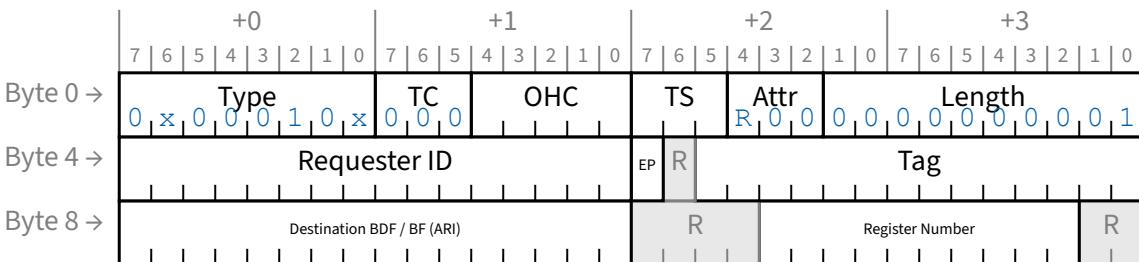


Figure 2-48 ↑↓Flit Mode↓ Configuration Request ↑↓Header - Flit Mode↑ §

2.2.8 Message Request Rules §

This document defines the following groups of Messages:

- INTx Interrupt Signaling
- Power Management
- Error Signaling
- Locked Transaction Support
- Slot Power Limit Support
- Vendor-Defined Messages
- Latency Tolerance Reporting (LTR) Messages
- Optimized Buffer Flush/Fill (OBFF) Messages
- Device Readiness Status (DRS) Messages
- Function Readiness Status (FRS) Messages
- Hierarchy ID Messages
- Precision Time Measurement (PTM) Messages
- Integrity and Data Encryption (IDE) Messages

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

- All Message Requests include the following fields in addition to the common header fields (see § [Figure 2-49](#) and § [Figure 2-50](#)):
 - Requester ID[15:0]
 - Message Code[7:0] - Indicates the particular Message embodied in the Request.
 - EP - For Messages with data only, indicates data payload is poisoned (see § [Section 2.7](#)); Reserved for Messages without data.
- All Message Requests use the Msg or MsgD TLP Type.
- The Message Code field must be fully decoded (Message aliasing is not permitted).
- The Attr[2] field is not Reserved unless specifically indicated as Reserved.
- Except as noted, the Attr[1:0] field is Reserved.
- Except as noted, TH is not applicable to Message Requests and the bit is Reserved.
- AT[1:0] must be 00b except for Routed by Address Messages in Flit Mode (see § [Table 2-20](#)). Receivers are not required or encouraged to check this.
- Bytes 8 through 15 are Reserved unless specifically defined.
- Bytes 8 through 15 must be copied intact during Translation between Flit Mode and Non-Flit Mode, regardless of Message Code.
- Byte 6, bits 6:0 must be copied intact during Translation between Flit Mode and Non-Flit Mode, regardless of Message Code.
- Message Requests are posted and do not require Completion.
- Message Requests follow the same ordering rules as Memory Write Requests.

Many types of Messages, including Vendor-Defined Messages, are potentially usable in non-D0 states, and it is strongly recommended that the handling of Messages by Ports be the same when the Port's Bridge Function is in D1, D2, and D3_{Hot} as it is in D0. It is strongly recommended that Type 0 Functions support the generation and reception of Messages in non-D0 states.

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": ["0", "x", "1"], "name": "Fmt", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "value": ["1", "0", "r2", "r1", "r0"], "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "value": ["T9"], "name": "T9", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "value": ["TC"], "name": "TC", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "value": ["T8"], "name": "T8", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "value": ["A2"], "name": "A2", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "value": ["TH"], "name": "TH", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "value": ["TD"], "name": "TD", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "value": ["EP"], "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "value": ["Attr"], "name": "Attr", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "value": ["AT"], "name": "AT", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "value": ["Length"], "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "value": ["Requester ID"], "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "value": ["{Reserved, except as noted}"], "name": "{Reserved, except as noted}", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 0, "value": ["Message Code"], "name": "Message Code", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 11, "msbit": 0, "value": ["{Except as noted, bytes 8 through 11 are reserved.}"], "name": "{Except as noted, bytes 8 through 11 are reserved.}", "addClass": "regFieldReservedText", "attr": "ro" }, { "lsbyte": 12, "lsbit": 7, "msbyte": 15, "msbit": 0, "value": ["{Except as noted, bytes 12 through 15 are reserved.}"], "name": "{Except as noted, bytes 12 through 15 are reserved.}", "addClass": "regFieldReservedText", "attr": "ro" } ] } ↓

```

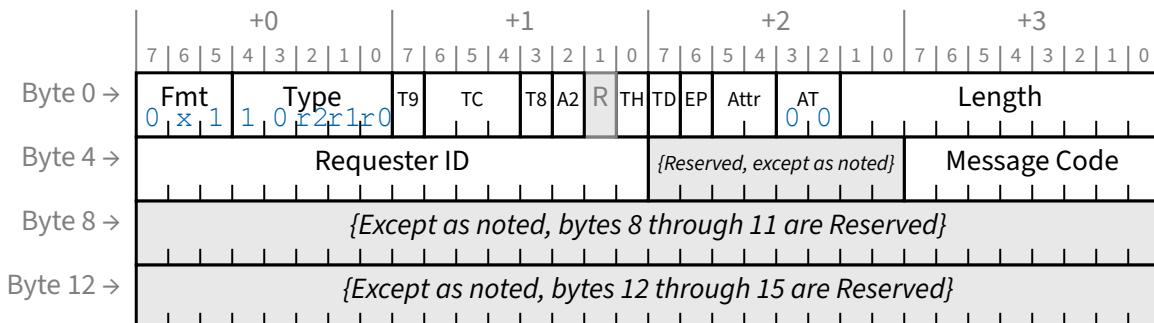


Figure 2-49 Message Request Header - Non-Flit Mode §

Base 6.4 vs Base 6.3

```

    ↓↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
      { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "value": "[0, \"x\", 1, \"1\", 0, \"r2\", \"r1\", \"r0\"], \"attr\": \"ro\"", "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro"}, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro"}, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro"}, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro"}, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro"}, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro"}, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 6, "name": "[Reserved, except as noted]", "addClass": "regFieldReservedText", "regFieldVerySmallText": "ro"}, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "attr": "ro"}, { "lsbyte": 8, "lsbit": 7, "msbyte": 11, "msbit": 0, "name": "[Except as noted, bytes 8 through 11 are reserved.]", "addClass": "regFieldReservedText", "attr": "ro"}, { "lsbyte": 12, "lsbit": 7, "msbyte": 15, "msbit": 0, "name": "[Except as noted, bytes 12 through 15 are reserved.]", "addClass": "regFieldReservedText", "attr": "ro" } ] }↓
  
```

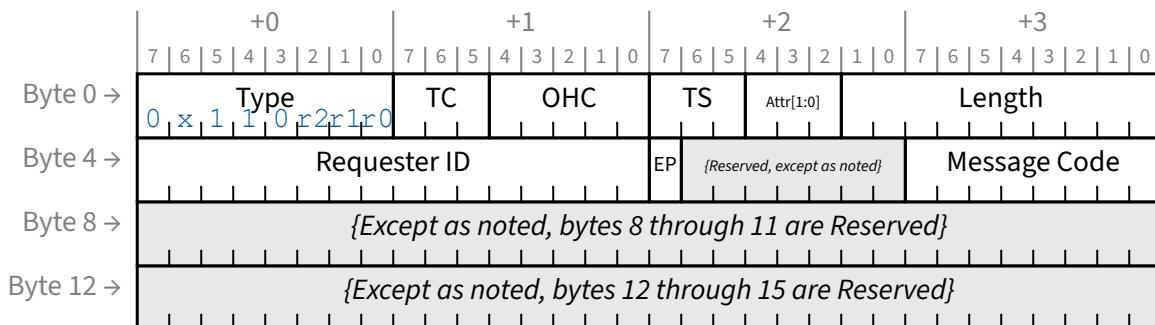


Figure 2-50 Message Request Header - Flit Mode §

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

- Message routing is determined using the r[2:0] sub-field of the Type field
 - Message Routing r[2:0] values are defined in § Table 2-20
 - Permitted values are defined in the following sections for each Message

Table 2-20 Message Routing §

| r[2:0] (b) | Description | Bytes 8 to 15 ²⁷ |
|------------|---|-----------------------------|
| 000 | Routed to Root Complex | Reserved |
| 001 | Routed by Address + AT , in Flit Mode ²⁸ | Address/AT |
| 010 | Routed by ID | See § Section 2.2.4.2 |
| 011 | Broadcast from Root Complex | Reserved |
| 100 | Local - Terminate at Receiver | Reserved |
| 101 | Gathered and routed to Root Complex ²⁹ | Reserved |

27. Except as noted, e.g., Vendor-Defined Messages.

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

29. This routing type is used only for PME_TO_Ack , and is described in § Section 5.3.3.2.1 .

| | | |
|------------|----------------------------------|---------------|
| r[2:0] (b) | Description | Bytes 8 to 15 |
| 110 to 111 | Reserved - Terminate at Receiver | Reserved |

In Flit Mode, when Route by ID is used and the Destination Segment is different from the Requester Segment, OHC-A4 must be present and include the Destination Segment in byte 0 and DSV must be Set. DSV is permitted to be Set when the Destination Segment is the same as the Requester Segment. DSV must be Clear when Route by ID is not used. When DSV is clear, the Destination Segment field must be set to 00h. OHC-A4 must be present for Route by ID Messages that require PASID. OHC-A1 must be present for Routed to Root Complex Messages that require PASID, ER or PMR.

2.2.8.1 INTx Interrupt Signaling - Rules §

A Message Signaled Interrupt (MSI or MSI-X) is the preferred interrupt signaling mechanism in PCI Express (see § Section 6.1). However, in some systems, there may be Functions that cannot support the MSI or MSI-X mechanisms, or it is possible that system firmware/software does not enable MSI or MSI-X. The INTx virtual wire interrupt signaling mechanism, when implemented, can be used to support cases where the MSI or MSI-X mechanisms cannot be used. Switches must support passing interrupts via this mechanism. The following rules apply to the INTx Interrupt Signaling mechanism:

- The INTx mechanism uses eight distinct Messages (see § Table 2-21).
- Assert_INTx/Deassert_INTx Messages do not include a data payload (TLP Type is Msg).
- The Length field is Reserved.
- With Assert_INTx/Deassert_INTx Messages, the Function Number field in the Requester ID must be 0. Note that the Function Number field is a different size for non-ARI and ARI Requester IDs.
- Assert_INTx/Deassert_INTx Messages are only issued by Upstream Ports.
 - Receivers may optionally check for violations of this rule. If a Receiver implementing this check determines that an Assert_INTx/Deassert_INTx violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).
- Assert_INTx and Deassert_INTx interrupt Messages must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).

Table 2-21 INTx Mechanism Messages §

| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support ³⁰ | | | | Description/Comments | |
|-------------|---------------|--------------------|-----------------------|----|----|----|---|--|
| | | | RC | Ep | Sw | Br | | |
| Assert_INTA | 0010 0000 | 100 | All: | | | | Assert INTA virtual wire Note: These Messages are used for Conventional PCI-compatible INTx emulation. | |
| | | | r | | tr | | | |
| | | | As Required: | | | | | |
| | | | | t | | t | | |
| Assert_INTB | 0010 0001 | 100 | All: | | | | Assert INTB virtual wire | |

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

Base 6.4 vs Base 6.3

| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
|---------------|------------------|-----------------------|--------------|----|----|----|----------------------------|
| | | | RC | Ep | Sw | Br | |
| | | | r | | tr | | As Required: |
| | | | | t | | t | |
| Assert_INTC | 0010 0010 | 100 | All: | | | | Assert INTC virtual wire |
| | | | r | | tr | | |
| | | | As Required: | | | | |
| | | | | t | | t | |
| Assert_INTD | 0010 0011 | 100 | All: | | | | Assert INTD virtual wire |
| | | | r | | tr | | |
| | | | As Required: | | | | |
| | | | | t | | t | |
| Deassert_INTA | 0010 0100 | 100 | All: | | | | Deassert INTA virtual wire |
| | | | r | | tr | | |
| | | | As Required: | | | | |
| | | | | t | | t | |
| Deassert_INTB | 0010 0101 | 100 | All: | | | | Deassert INTB virtual wire |
| | | | r | | tr | | |
| | | | As Required: | | | | |
| | | | | t | | t | |
| Deassert_INTC | 0010 0110 | 100 | All: | | | | Deassert INTC virtual wire |
| | | | r | | tr | | |
| | | | As Required: | | | | |
| | | | | t | | t | |
| Deassert_INTD | 0010 0111 | 100 | All: | | | | Deassert INTD virtual wire |
| | | | r | | tr | | |
| | | | As Required: | | | | |
| | | | | t | | t | |

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

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

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

- INTx Interrupt Signaling is disabled when the Interrupt Disable bit of the Command register (see § [Section 7.5.1.1.3](#)) is Set.
 - Any INTx virtual wires that are active when the Interrupt Disable bit is Set must be deasserted by transmitting the appropriate Deassert_INTx Message(s).
- Virtual and actual PCI to PCI Bridges must map the virtual wires tracked on the secondary side of the Bridge according to the Device Number of the device on the secondary side of the Bridge, as shown in § [Table 2-22](#).
- Switches must track the state of the four virtual wires independently for each Downstream Port, and present a “collapsed” set of virtual wires on its Upstream Port.
- If a Switch Downstream Port goes to DL_Down status, the INTx virtual wires associated with that Port must be deasserted, and the Switch Upstream Port virtual wire state updated accordingly.
 - If this results in deassertion of any Upstream INTx virtual wires, the appropriate Deassert_INTx Message(s) must be sent by the Upstream Port.
- The Root Complex must track the state of the four INTx virtual wires independently for each of its Downstream Ports, and map these virtual signals to system interrupt resources.
 - Details of this mapping are system implementation specific.
- If a Downstream Port of the Root Complex goes to DL_Down status, the INTx virtual wires associated with that Port must be deasserted, and any associated system interrupt resource request(s) must be discarded.

Table 2-22 Bridge Mapping for INTx Virtual Wires §

| Requester ID[7:3] from the Assert_INTx/Deassert_INTx Message received on Secondary Side of Bridge (Interrupt Source ³¹) If ARI Forwarding is enabled, the value 0 must be used instead of Requester ID[7:3]. | INTx Virtual Wire on Secondary Side of Bridge | Mapping to INTx Virtual Wire on Primary Side of Bridge |
|---|---|--|
| 0,4,8,12,16,20,24,28 | INTA | INTA |
| | INTB | INTB |
| | INTC | INTC |
| | INTD | INTD |
| 1,5,9,13,17,21,25,29 | INTA | INTB |
| | INTB | INTC |
| | INTC | INTD |

31. 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.

| Requester ID[7:3] from the Assert_INTx/Deassert_INTx Message received on Secondary Side of Bridge (Interrupt Source) If ARI Forwarding is enabled, the value 0 must be used instead of Requester ID[7:3]. | INTx Virtual Wire on Secondary Side of Bridge | Mapping to INTx Virtual Wire on Primary Side of Bridge |
|--|---|--|
| 2,6,10,14,18,22,26,30 | INTD | INTA |
| | INTA | INTC |
| | INTB | INTD |
| | INTC | INTA |
| | INTD | INTB |
| 3,7,11,15,19,23,27,31 | INTA | INTD |
| | INTB | INTA |
| | INTC | INTB |
| | INTD | INTC |

Base 6.4 vs Base 6.3

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 2-22 is applied hierarchically at every Switch. In addition, PCI Express/PCI and PCI/PCI Bridges perform a similar mapping function.

IMPLEMENTATION NOTE: VIRTUAL WIRE MAPPING FOR INTX INTERRUPTS FROM ARI DEVICES §

The implied Device Number for an ARI Device is 0. When ARI-aware software (including BIOS and operating system) enables ARI Forwarding in the Downstream Port immediately above an ARI Device in order to access its Extended Functions, software must comprehend that the Downstream Port will use Device Number 0 for the virtual wire mappings of INTx interrupts coming from all Functions of the ARI Device. If non-ARI-aware software attempts to determine the virtual wire mappings for Extended Functions, it can come up with incorrect mappings by examining the traditional Device Number field and finding it to be non-0.

2.2.8.2 Power Management Messages §

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

- § Table 2-23 defines the Power Management Messages.
- Power Management Messages do not include a data payload (TLP Type is Msg).
- The Length field is Reserved.
- With PM_Active_State_Nak Messages, the Function Number field in the Requester ID must contain the Function Number of the Downstream Port that sent the Message, or else 000b for compatibility with earlier revisions of this specification.
- With PME_TO_Ack Messages, the Function Number field in the Requester ID must be Reserved, or else for compatibility with earlier revisions of this specification must contain the Function Number of one of the Functions associated with the Upstream Port. Note that the Function Number field is a different size for non-ARI and ARI Requester IDs.
- Power Management Messages must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).

Table 2-23 Power Management Messages §

| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments | |
|---------------------|------------------|-----------------------|------------------------------------|----|----|----|--|--|
| | | | RC | Ep | Sw | Br | | |
| PM_Active_State_Nak | 0001 0100 | 100 | t | r | tr | r | Terminate at Receiver | |
| PME_PME | 0001 1000 | 000 | All: | | | | Sent Upstream by PME-requesting component. Propagates Upstream. | |
| | | | r | | tr | t | | |
| | | | If PME supported: | | | | | |
| | | | | t | | | | |
| PME_Turn_Off | 0001 1001 | 011 | t | r | | r | Broadcast Downstream | |
| PME_TO_Ack | 0001 1011 | 101 | r | t | | t | Sent Upstream by Upstream Port. See § Section 5.3.3.2.1. | |
| | | | (Note: Switch handling is special) | | | | | |

2.2.8.3 Error Signaling Messages §

Error Signaling Messages are used to signal errors that occur on specific transactions and errors that are not necessarily associated with a particular transaction. These Messages are initiated by the agent that detected the error.

- § Table 2-24 defines the Error Signaling Messages.
- Error Signaling Messages do not include a data payload (TLP Type is Msg).
- The Length field is Reserved.
- With Error Signaling Messages, the Function Number field in the Requester ID must indicate which Function is signaling the error. Note that the Function Number field is a different size for non-ARI and ARI Requester IDs.

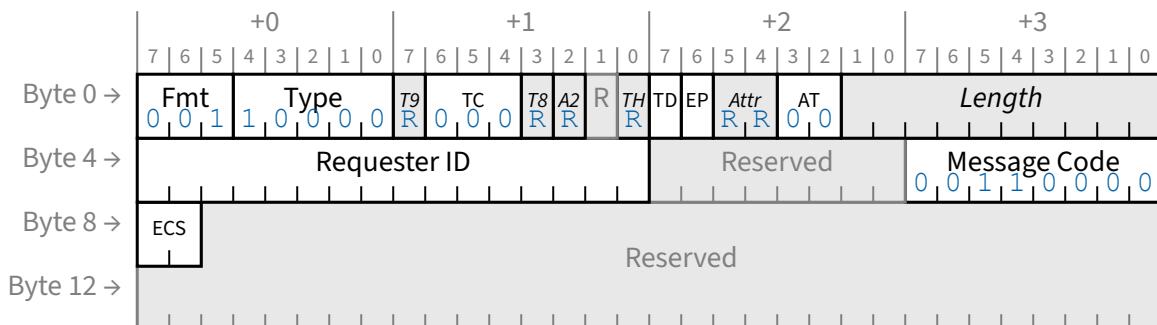
- Error Signaling Messages must use the default Traffic Class designator (TC0) Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § [Section 6.2](#)).

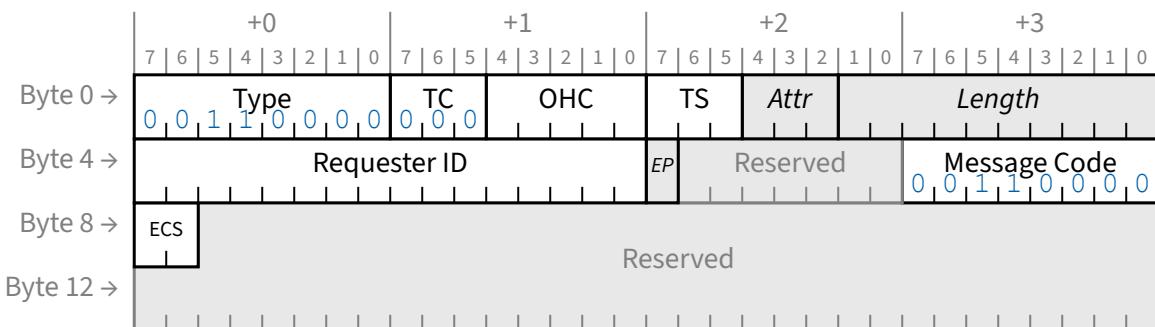
Table 2-24 Error Signaling Messages §

| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
|---------------------|------------------|-----------------------|---------|----|----|----|---|
| | | | RC | Ep | Sw | Br | |
| ERR_COR | 0011 0000 | 000 | r | t | tr | t | This Message is issued when the Function or Device detects a correctable error on the PCI Express interface. |
| ERR_NONFATAL | 0011 0001 | 000 | r | t | tr | t | This Message is issued when the Function or Device detects a Non-Fatal, uncorrectable error on the PCI Express interface. |
| ERR_FATAL | 0011 0011 | 000 | r | t | tr | t | This Message is issued when the Function or Device detects a Fatal, uncorrectable error on the PCI Express interface. |

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

- ERR_COR Messages have an ERR_COR Subclass (ECS) field in the Message header that enables different subclasses to be distinguished from each other. See § [Figure 2-51](#) ↑ and § [Figure 2-52](#) ↑ . ERR_NONFATAL and ERR_FATAL Messages do not have the ECS field.

*Figure 2-51 ERR_COR Message - Non-Flit Mode* §

Figure 2-52 **ERR_COR Message - Flit Mode** §

- The **ERR_COR Subclass (ECS)** field is encoded as shown in § Table 2-25, indicating the **ERR_COR Message subclass**.

Table 2-25 **ERR_COR Subclass (ECS) Field Encodings** §

| ECS Coding ↑↑(b)↑ | Description |
|----------------------|--|
| 00 | ECS Legacy - The value inherently used if a Requester does not support ECS capability. ECS-capable Requesters must not use this value. See ↓↓see↓ § Section 7.5.3.3. |
| 01 | ECS SIG_SFW - Must be used by an ECS-capable Requester when signaling a <u>DPC</u> or <u>SFI</u> event with an <u>ERR_COR</u> Message. |
| 10 | ECS SIG_OS - Must be used by an ECS-capable Requester when signaling an <u>AER</u> or <u>RP PIO</u> event with an <u>ERR_COR</u> Message. |
| 11 | ECS Extended - Intended for possible future use. Requesters must not use this value. Receivers must handle the signal internally the same as <u>ECS SIG_OS</u> . |

2.2.8.4 Locked Transactions Support §

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

- § Table 2-26 defines the Unlock Messages.
- The Unlock Message does not include a data payload (TLP Type is Msg).
- The Length field is Reserved.
- With Unlock Messages, the Function Number field in the Requester ID is Reserved.
- The Unlock Message must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).

Table 2-26 Unlock Message §

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

2.2.8.5 Slot Power Limit Support §

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

- § Table 2-27 defines the Set_Slot_Power_Limit Message .
- The Set_Slot_Power_Limit Message includes a 1 DW data payload (TLP Type is MsgD).
- The Set_Slot_Power_Limit Message must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).

Table 2-27 Set_Slot_Power_Limit Message §

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

The Set_Slot_Power_Limit Message includes a one DW data payload. The data payload is copied from the Slot Capabilities register of the Downstream Port and is written into the Device Capabilities register of the Upstream Port on the other side of the Link. Bits 1:0 of Byte 1 of the data payload map to the Slot Power Limit Scale field and bits 7:0 of Byte 0 map to the Slot Power Limit Value field. Bits 7:0 of Byte 3, 7:0 of Byte 2, and 7:2 of Byte 1 of the data payload must all be set to zero by the Transmitter and ignored by the Receiver. This Message must be sent automatically by the Downstream Port (of a Root Complex or a Switch) when one of the following events occurs:

- On a Configuration Write to the Slot Capabilities register (see § Section 7.5.3.9) when the Data Link Layer reports DL_Up status.
- Any time when a Link transitions from a non- DL_Up status to a DL_Up status (see § Section 2.9.2) and the Auto Slot Power Limit Disable bit is Clear in the Slot Control Register. This transmission is optional if the Slot Capabilities register has not yet been initialized.

The component on the other side of the Link (with Endpoint, Switch, or Bridge Functions) that receives Set_Slot_Power_Limit Message must copy the values in the data payload into the Device Capabilities register associated with the component's Upstream Port. PCI Express components that are targeted exclusively for integration on the system planar (e.g., system board) as well as components that are targeted for integration on an adapter where power consumption of the entire adapter is below the lowest power limit specified for the adapter form factor (as defined in the corresponding form factor specification) are permitted to hardwire the value of all 0's in the Captured Slot Power Limit Scale and Captured Slot Power Limit Value fields of the Device Capabilities Register , and are not required to copy the Set_Slot_Power_Limit Message payload into that register.

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

2.2.8.6 Vendor-Defined Messages §

The Vendor-Defined Messages allow expansion of PCI Express messaging capabilities, either as a general extension to [PCIe] or a vendor-specific extension. This section defines the rules associated with these Messages generically.

- The Vendor-Defined Messages (see § Table 2-28) use the header format shown in [Figure 2-53 and § Figure 2-54↑](#).
 - The Requester ID is implementation specific. The Requester ID field *MUST* contain the value associated with the Requester.³²
 - If the Route by ID routing is used, bytes 8 and 9 form a 16-bit field for the destination ID
 - otherwise these bytes are Reserved.
 - Bytes 10 and 11 form a 16-bit field for the Vendor ID, as defined by PCI-SIG®, of the vendor defining the Message.
 - Bytes 12 through 15 are available for vendor definition.
 - The low 7 bits of byte 6 is available for vendor definition. Byte 6, bit 7 is Reserved in Non-Flit Mode and is the EP bit in Flit Mode.

Table 2-28 Vendor-Defined Messages §

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

[Notes:↑](#)

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

32. ACS Source Validation (see § Section 6.12.1.1) checks the Requester ID on all Requests, including Vendor-Defined Messages. This validation depends on the Requester ID properly identifying the Requester.

Base 6.4 vs Base 6.3

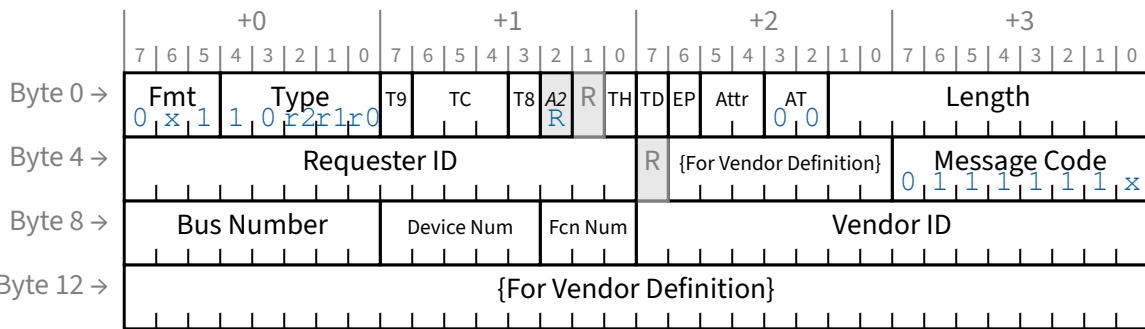


Figure 2-53 Header for Vendor-Defined Messages - Non-Flit Mode §

```
↓↓[ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [ {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "value": ["0", "x", "1", "1", "0", "r2", "r1", "r0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro"}, {"lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro"}, {"lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "[For Vendor Definition]", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": ["0", "1", "1", "1", "1", "1", "1", "x"], "attr": "ro"}, {"lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination ID if ID Routed; otherwise Reserved", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "Vendor ID", "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "[For Vendor Definition]", "attr": "ro"} ] ]↓
```

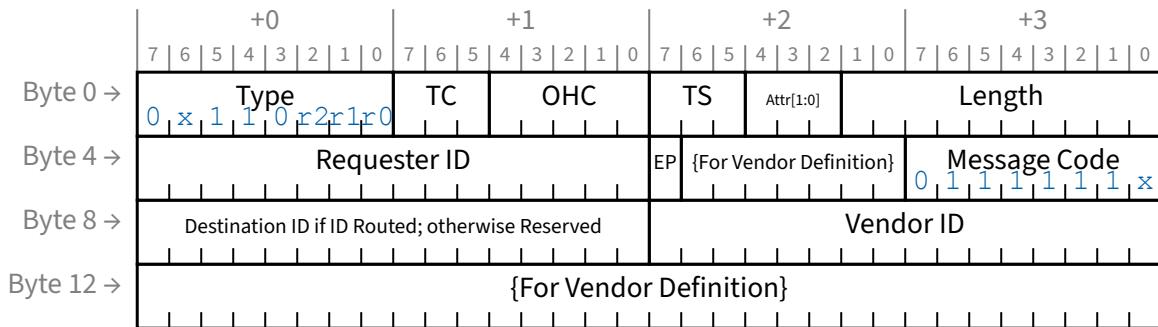


Figure 2-54 Header for Vendor-Defined Messages - Flit Mode §

- A data payload may be included with either type of Vendor-Defined Message (TLP type is Msg if no data payload is included or MsgD if a data payload is included).
- For both types of Vendor-Defined Messages, the Attr[1:0] and Attr[2] fields are not Reserved.
- Messages defined by different vendors or by PCI-SIG are distinguished by the value in the Vendor ID field.
 - The further differentiation of Messages defined by a particular vendor is beyond the scope of this document.
 - Support for Messages defined by a particular vendor is implementation specific, and beyond the scope of this document.

- Completers silently discard Vendor-Defined Type 1 Messages that they are not designed to receive - this is not an error condition.
 - When an ID Routed Message targeting a Function that is not implemented is detected, it is implementation specific whether that message is silently discarded or signals Unsupported Request.
- Completers handle the receipt of an unsupported Vendor-Defined Type 0 Message as an Unsupported Request, and the error is reported according to § Section 6.2 .

[PCIe-to-PCI-PCI-X-Bridge] defines additional requirements for Vendor-Defined Messages that are designed to be interoperable with PCI-X Device ID Messages. This includes restrictions on the contents of the Tag[7:0] field and the Length[9:0] field as well as specific use of Bytes 12 through 15 of the message header. Vendor-Defined Messages intended for use solely within a PCI Express environment (i.e., not intended to address targets behind a PCI Express to PCI/PCI-X Bridge) are not subject to the additional rules. Refer to [PCIe-to-PCI-PCI-X-Bridge] for details. Refer to § Section 2.2.6.2 for considerations regarding larger-Tag capabilities.

2.2.8.6.1 PCI-SIG Defined VDMs §

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

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

- PCI-SIG-Defined VDMs use the Header format shown in [\\$ Figure 2-55 and § Figure 2-56](#) .
- The Requester ID field must contain the value associated with the Requester.
- The Message Code must be [01111111b](#).[0111 1111b](#).
- The Vendor ID must be 0001h, which is assigned to the PCI-SIG.
- The Subtype field distinguishes the specific PCI-SIG-Defined VDMs. See § Appendix F. for a list of PCI-SIG-Defined VDMs.

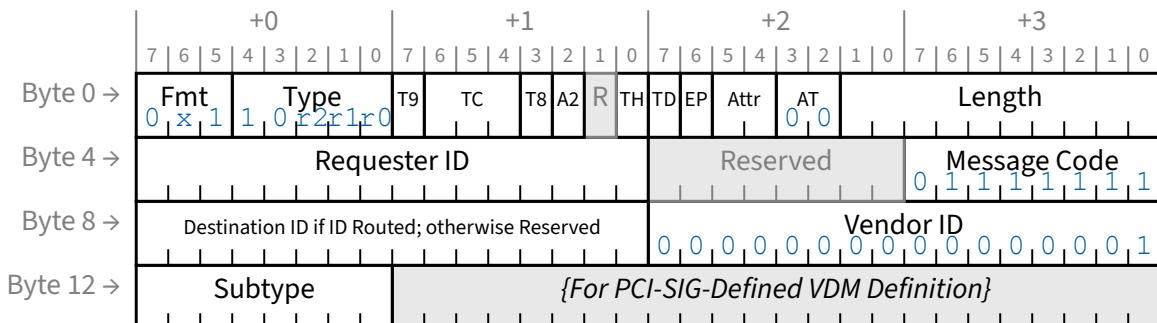


Figure 2-55 Header for PCI-SIG-Defined VDMs - Non-Flit Mode §

Base 6.4 vs Base 6.3

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "value": ["0", "x", "1", "1", "0", "r2", "r1", "r0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 6, "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "Message Code", "value": ["0", "1", "1", "1", "1", "1", "1"], "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination ID if ID Routed; otherwise Reserved", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "Vendor ID", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro" }, { "lsbyte": 12, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "Subtype", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 13, "msbit": 7, "name": "[For PCI-SIG-Defined VDM Definition]", "attr": "ro" } ] } ↓
```

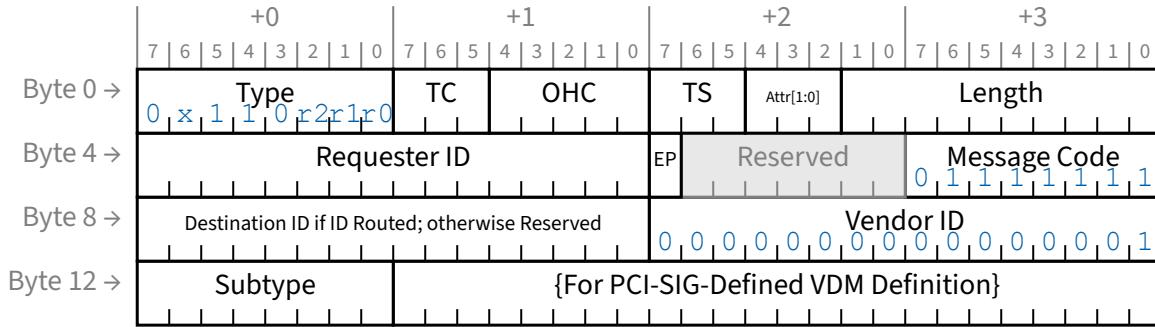


Figure 2-56 Header for PCI-SIG-Defined VDMs - Flit Mode §

2.2.8.6.2 Device Readiness Status (DRS) Message §

The Device Readiness Status (DRS) protocol (see § Section 6.22.1) uses the PCI-SIG-Defined VDM mechanism (see § Section 2.2.8.6.1). The DRS Message is a PCI-SIG-Defined VDM (Vendor-Defined Type 1 Message) with no **↑↓data↑** payload.

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

- § Table 2-29 **↑↓, § Figure 2-57,↑** and **↑↓§ Figure 2-58↑** illustrate and define the DRS Message.
- The TLP Type must be Msg.
- The TC[2:0] field must be 000b.
- The Attr[2:0] field is Reserved.
- The Tag field is Reserved.
- The Subtype field must be 08h.
- The Message Routing field must be **↑↓set to 100b - Local - Terminate at Receiver.**

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

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

Base 6.4 vs Base 6.3

Table 2-29 DRS Message §

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

The format of the DRS Message is shown in Figure 2-57 and Figure 2-58.

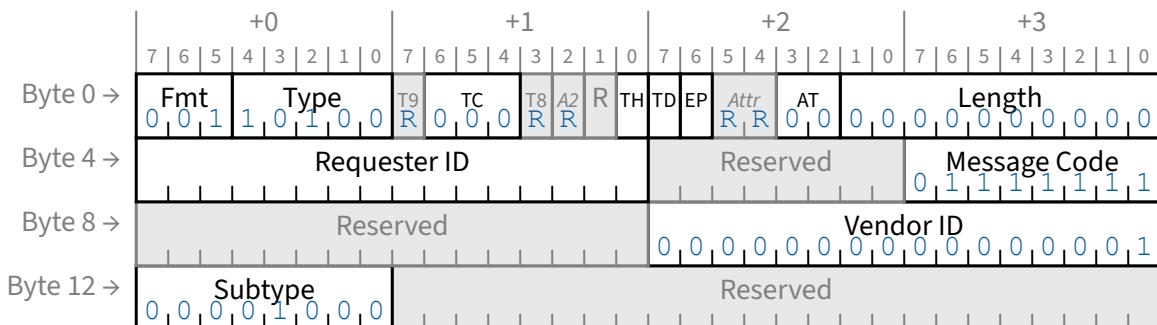


Figure 2-57 DRS Message - Non-Flit Mode §

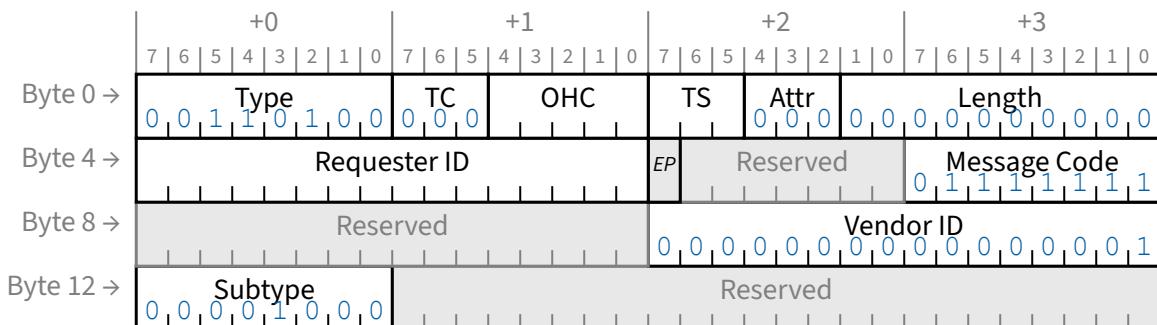


Figure 2-58 DRS Message - Flit Mode §

2.2.8.6.3 Function Readiness Status Message (FRS Message)

The Function Readiness Status (FRS) protocol (see § [Section 6.22.2](#)) uses the PCI-SIG-Defined VDM mechanism (see § [Section 2.2.8.6.1](#)). The FRS message is a PCI-SIG-Defined VDM (Vendor-Defined Type 1 Message) with no [↑↑data↑↑](#) payload.

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

- § Table 2-30 [↑↑, § Figure 2-59, ↑](#) and [↑↑§ Figure 2-60↑](#) illustrate and define the FRS Message.
 - The TLP Type must be Msg.
 - The TC[2:0] field must be 000b.
 - The Attr[2:0] field is Reserved.

- The Tag field is Reserved.
- The Subtype field must be 09h.
- The FRS Reason[3:0] field indicates why the FRS Message was ↓↓generated↓↓ ↑↑generated. Encodings are:↑↑

0001b: DRS Message Received

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

0010b: D3_{Hot} to D0 Transition Completed

A D3_{Hot} to D0 transition has completed, and the Function indicated by the Message Requester ID is now Configuration-Ready and has returned to the D0_{uninitialized} or D0_{active} state depending on the setting of the No_Soft_Reset bit (see § Section 7.5.2.2)

0011b: FLR Completed

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

1000b: VF Enabled

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

1001b: VF Disabled

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

Others:

All other ↓↓values↓↓ ↑↑encodings are:↑↑ Reserved

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

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

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

Table 2-30 FRS Message §

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

The format of the FRS Message is shown in ↑↑§ Figure 2-59↑↑ and § Figure 2-60 below:

Base 6.4 vs Base 6.3

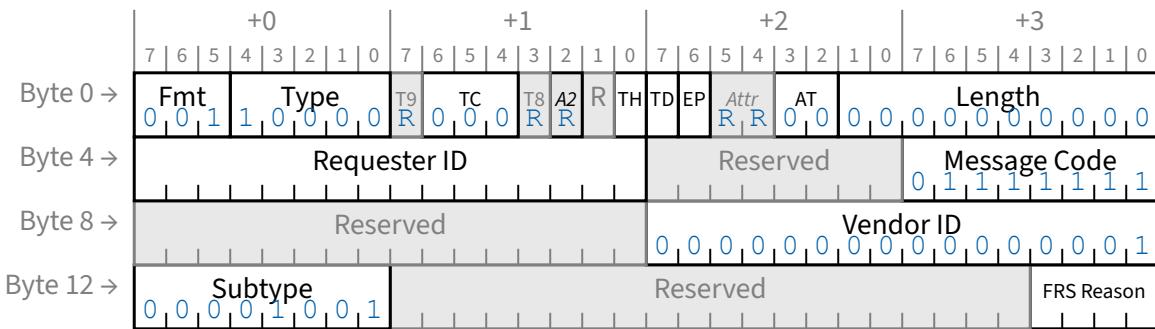


Figure 2-59 FRS Message - Non-Flit Mode §

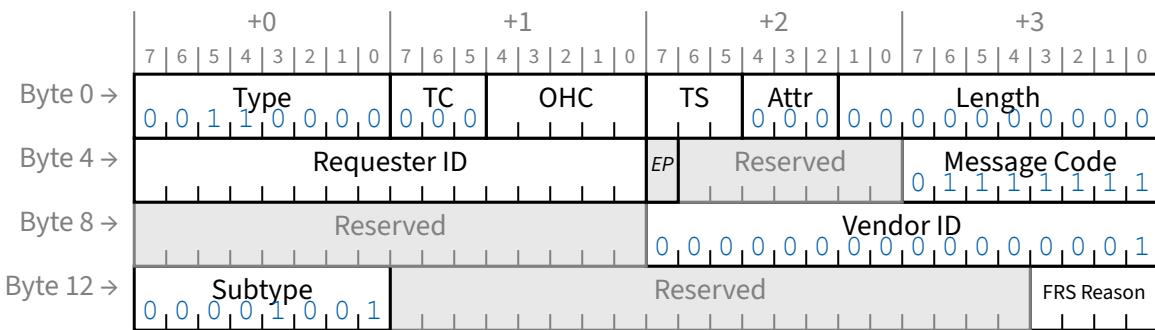


Figure 2-60 FRS Message - Flit Mode §

2.2.8.6.4 Hierarchy ID Message §

Hierarchy ID uses the PCI-SIG-Defined VDM mechanism (see § Section 2.2.8.6.1). The Hierarchy ID Message is a PCI-SIG-Defined VDM (Vendor-Defined Type 1 Message) with payload (MsgD).

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

- § Table 2-31 [↑](#), § Figure 2-61, [↑](#) and [↑](#) § Figure 2-62 [↑](#) illustrate and define the Hierarchy ID Message.
- The TLP Type must be MsgD.
- Each Message must include a 4-DWORD data payload.
- The Length field must be 4.
- The TC[2:0] field must be 000b.
- The Attr[2:0] field is Reserved.
- The Tag field is Reserved.
- The Subtype field is 01h.
- The Message Routing field must be 011b - Broadcast from Root Complex.

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

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

The payload of each Hierarchy ID Message contains the lower 128-bits of the System GUID.

For details of the Hierarchy ID, GUID Authority ID, and System GUID fields see § Section 6.25 .

Table 2-31 Hierarchy ID Message §

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

The format of the Hierarchy ID Message is shown in Figure 2-61 and Figure 2-62 below:

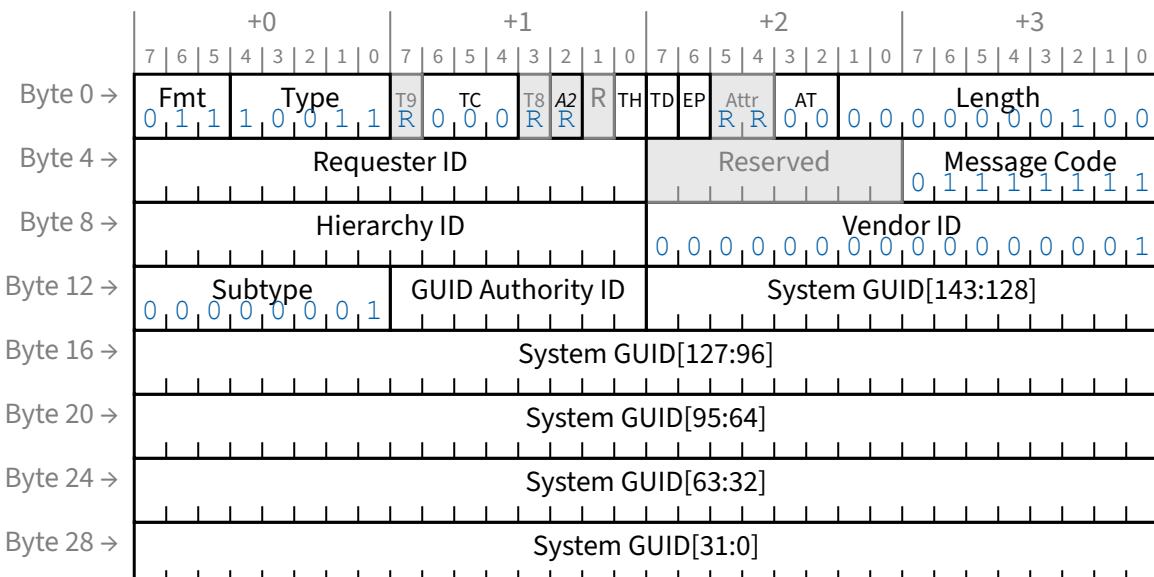


Figure 2-61 Hierarchy ID Message - Non-Flit Mode §

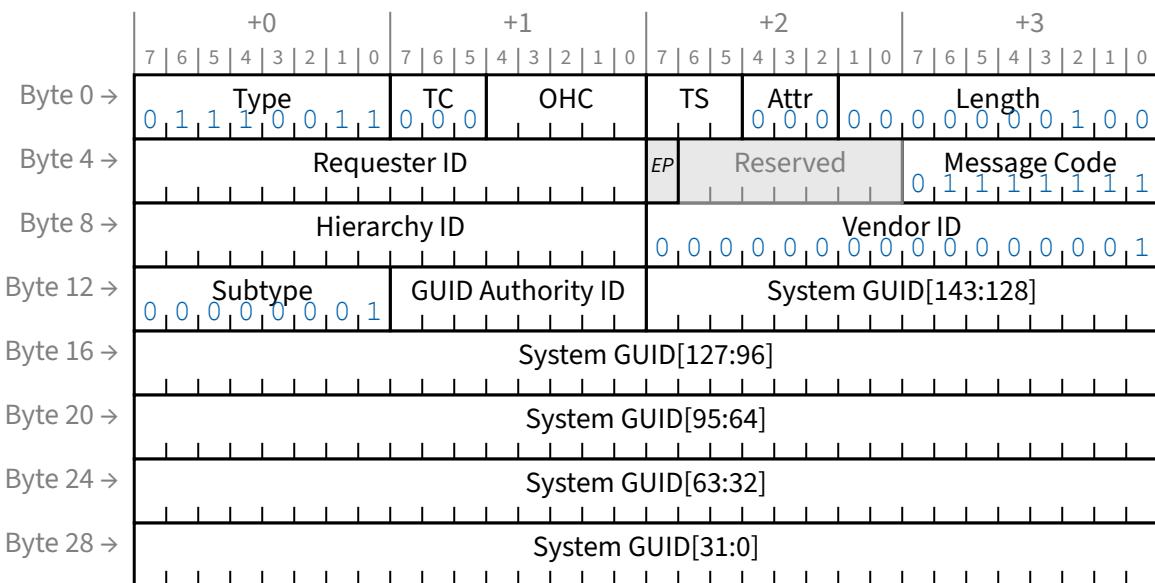


Figure 2-62 Hierarchy ID Message - Flit Mode §

2.2.8.7 Ignored Messages §

The messages listed in § Table 2-32 were previously used for a mechanism (Hot-Plug Signaling) that is no longer supported. Transmitters *MUST*@FLIT not transmit these messages. If message transmission is implemented, it must conform to the requirements of [PCIe-1.0a].

Beyond normal Link-Layer processing and mandatory checking for properly-formed TLPs, Receivers *MUST*@FLIT not process these messages further (i.e., carry out their originally architected Transaction-Layer functionality). If complete processing of these messages is implemented, Receivers must process these messages in conformance with the requirements [PCIe-1.0a].

Ignored messages listed in § Table 2-32 are handled by the Receiver as follows:

- The Physical and Data Link Layers must handle these messages identical to handling any other TLP.
- The Transaction Layer must account for flow control credit but take no other action in response to these messages.

Table 2-32 Ignored Messages §

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

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

2.2.8.8 Latency Tolerance Reporting (LTR) Message §

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

- § Table 2-33 [11↑](#), [§ Figure 2-63](#), and [§ Figure 2-64↑](#) defines the LTR Message.
- The LTR Message does not include a data payload (the TLP Type is Msg).
- The Length field is Reserved.
- The LTR Message must use the default Traffic Class designator (TC0). Receivers that implement LTR support must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).

Table 2-33 LTR Message §

| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support ¹ | | | | Description/Comments |
|------|---------------|--------------------|----------------------|----|----|----|-----------------------------|
| | | | RC | Ep | Sw | Br | |
| LTR | 0001 0000 | 100 | r | t | tr | | Latency Tolerance Reporting |

Notes:

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

```
↑↓[{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": ["0", "0", "1"], "name": "Fmt", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "value": ["1", "0", "1", "0", "0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldSmallText", "value": ["0", "0", "0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": ["0", "0", "0", "1", "0", "0", "0", "0"], "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 12, "lsbit": 7, "msbyte": 13, "msbit": 0, "name": "No Snoop Latency", "attr": "ro" }, { "lsbyte": 14, "lsbit": 7, "msbyte": 15, "msbit": 0, "name": "Snoop Latency", "attr": "ro" } ] }↓
```

Base 6.4 vs Base 6.3

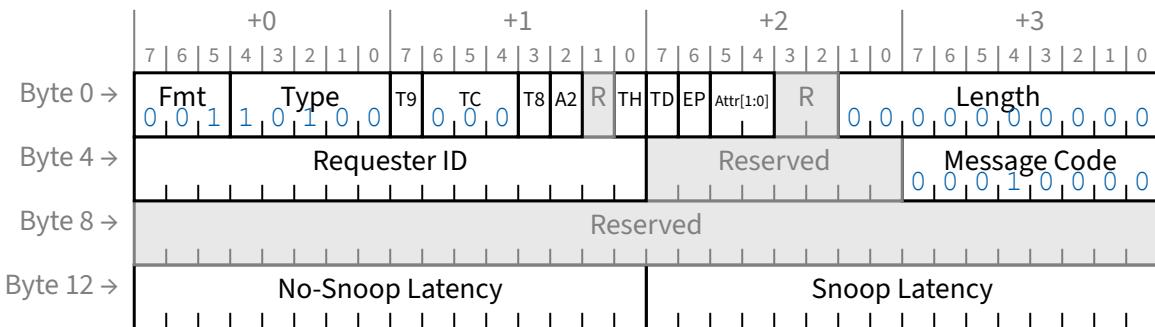


Figure 2-63 LTR Message - Non-Flit Mode §

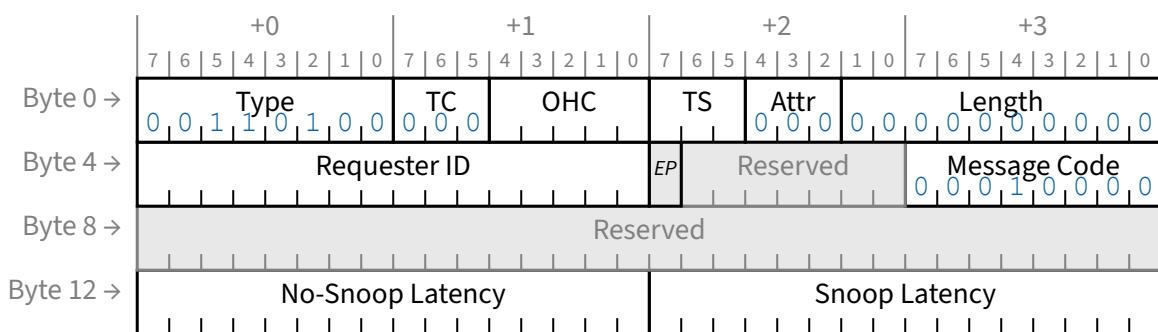


Figure 2-64 LTR Message - Flit Mode §

2.2.8.9 Optimized Buffer Flush/Fill (OBFF) Message §

The OBFF Message is optionally used to report platform central resource states to Endpoints. This mechanism is described in detail in § [Section 6.19](#).

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

- § [Table 2-34](#), § [Figure 2-65](#), and § [Figure 2-66](#) defines the OBFF Message.
- The OBFF Message does not include a data payload (TLP Type is Msg).
- The Length field is Reserved.
- The Requester ID must be set to the Transmitting Port's ID.
- The OBFF Message must use the default Traffic Class designator (TC0). Receivers that implement OBFF support must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § [Section 6.2](#)).

Base 6.4 VS Base 6.3

Table 2-34 OBFF Message §

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

Notes:

1. Support for OBFF is optional. Functions that support OBFF must implement the reporting and enable mechanisms described in § Chapter 7. [¶↓, Software Initialization and Configuration.](#) [↑↑.↑](#)

```
↑↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [{"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": [0, 0, 1], "name": "Fmt", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "value": [1, 0, 1, 0], "name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "value": [T9, 0, 0, 0, 0, 0, 0, 0], "name": "Requester ID", "attr": "ro"}, {"lsbyte": 1, "lsbit": 1, "msbyte": 1, "msbit": 3, "value": [TC, 0, 0, 0], "name": "TC", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 3, "value": [T8, A2, 0, 0], "name": "T8", "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 1, "value": [R, TH, TD, EP], "name": "R", "attr": "ro"}, {"lsbyte": 1, "lsbit": 4, "msbyte": 1, "msbit": 1, "value": [Attr[1:0], 0, 0, 0], "name": "Attr", "attr": "ro"}, {"lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 1, "value": [0, 0, 0, 0, 0, 0, 0, 0], "name": "Length", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 1, "value": [0, 0, 0, 0, 0, 0, 0, 0], "name": "Reserved", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 1, "value": [0, 0, 0, 0, 0, 0, 0, 0], "name": "Message Code", "attr": "ro"}]}↓
```

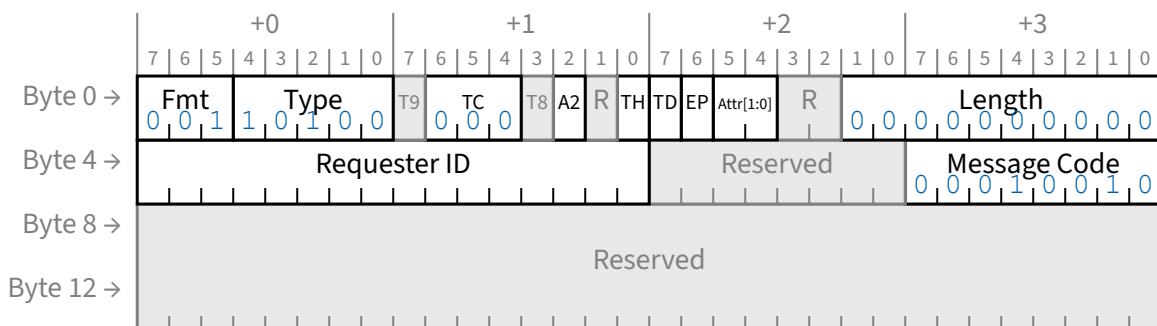


Figure 2-65 OBFF Message - Non-Flit Mode §

```

↑↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "Reserved", "fields": [{"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "value": ["0", "0", "1", "1", "0", "1", "0", "0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "value": ["0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro"}, {"lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "value": ["0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro"}, {"lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText regFieldReservedText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": ["0", "0", "1", "0", "0", "1", "0"], "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 3, "name": "OBFF Code", "addClass": "regFieldSmallText", "attr": "ro"}]}↓

```

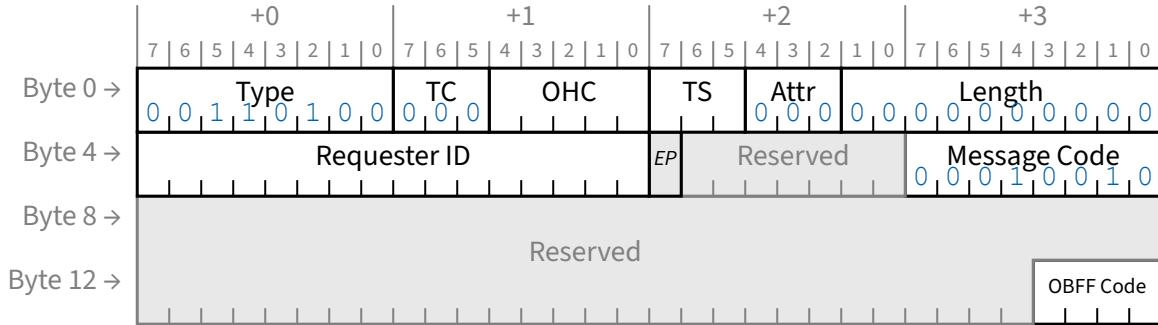


Figure 2-66 OBFF Message - Flit Mode §

2.2.8.10 Precision Time Measurement (PTM) Messages §

§ Table 2-35 defines the PTM Messages.

- The PTM Request and PTM Response Messages must use a TLP Type of Msg, and must not include a data payload. The Length field is reserved.
 - Figure 2-67 illustrates the format of the PTM Request and Response Messages.
- The PTM ResponseD Message must use a TLP Type of MsgD, and must include a 64 bit PTM Master Time field in bytes 8 through 15 of the TLP header and a 1 DW data payload containing the 32 bit Propagation Delay field.
 - Figure 2-68 illustrates the format of the PTM ResponseD Message.
 - Refer to Section 6.21.3.2 for details regarding how to populate the PTM ResponseD Message.
- The Requester ID must be set to the Transmitting Port's ID.
- A PTM dialog is defined as a matched pair of messages consisting of a PTM Request and the corresponding PTM Response or PTM ResponseD message.
- The PTM Messages must use the default Traffic Class designator (TC0). Receivers implementing PTM must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see Section 6.2).

Table 2-35 Precision Time Measurement Messages §

| Name | TLP Type | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
|---------------|----------|------------------|-----------------------|---------|----|----|----|--|
| | | | | RC | EP | Sw | Br | |
| PTM Request | Msg | 0101 0010 | 100 | r | t | tr | | Initiates PTM dialog |
| PTM Response | Msg | 0101 0011 | 100 | t | r | tr | | Completes current PTM dialog - does not carry timing information |
| PTM ResponseD | MsgD | 0101 0011 | 100 | t | r | tr | | Completes current PTM dialog - carries timing information |

```
]}], [{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [{"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": ["0", "0", "1"], "name": "Fmt", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "value": ["1", "0", "1", "0", "0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldSmallText", "value": ["0", "0", "0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "R", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Reserved", "isUnused": true, "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Message Code", "value": ["0", "1", "0", "1", "0", "0", "1", "X"], "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro"}]]
```

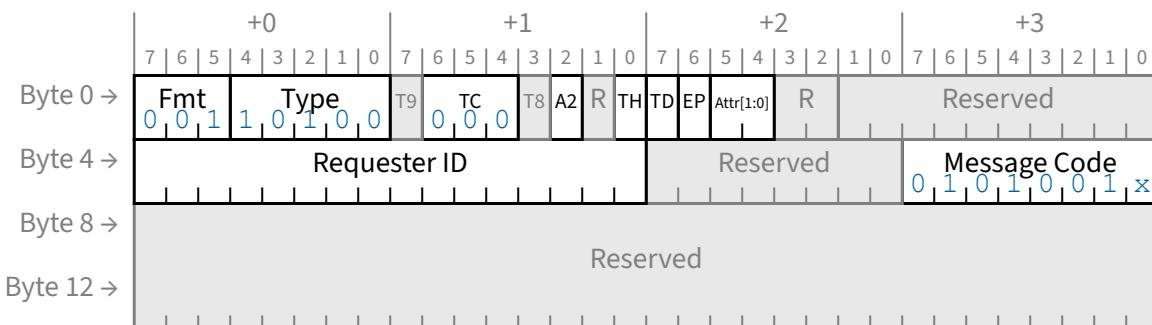


Figure 2-67 PTM Request/Response Message - Non-Flit Mode §

```
↓↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 160, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": [ "0", "1", "1" ], "name": "Fmt", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "value": [ "1", "0", "1", "0", "0" ], "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldSmallText", "value": [ "0", "0", "0" ], "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 2, "name": "TD2", "addClass": "regFieldSmallText", "attr": "ro" } ] }
```

```
"msbit": 6, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "R", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": [0, 0, 0, 0, 0, 0, 0, 1], "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": [0, 1, 0, 1, 0, 0, 1, 1], "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "PTM Master Time [63:32]", "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "PTM Master Time [31:0]", "attr": "ro"}, {"lsbyte": 19, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "Propagation Delay [31:0]", "attr": "ro"}]} ] } } ] }
```

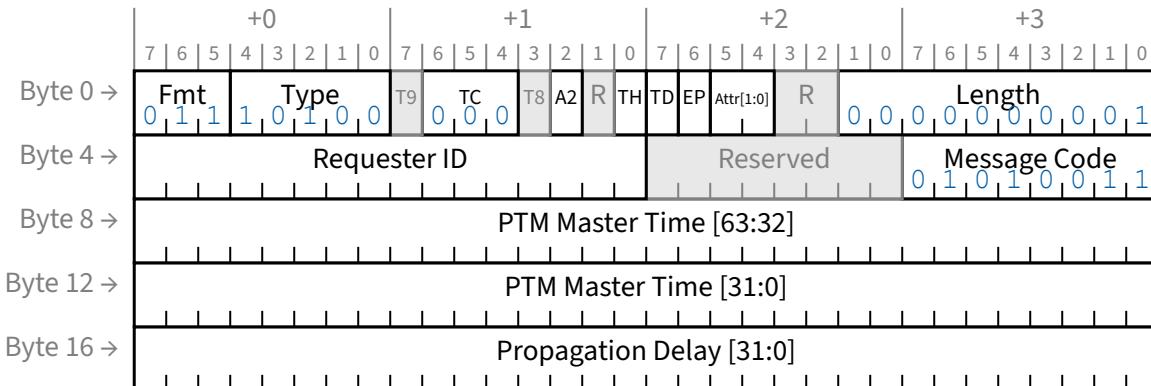


Figure 2-68 PTM ResponseD Message - Non-Flit Mode §

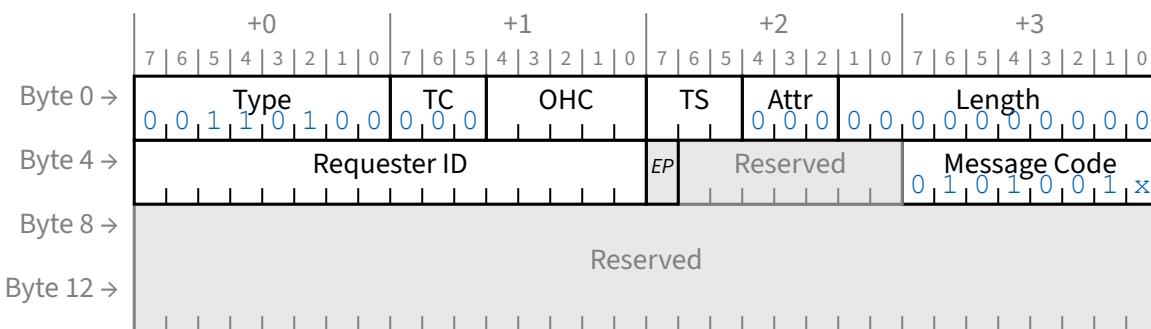


Figure 2-69 PTM Request/Response Message - Flit Mode §

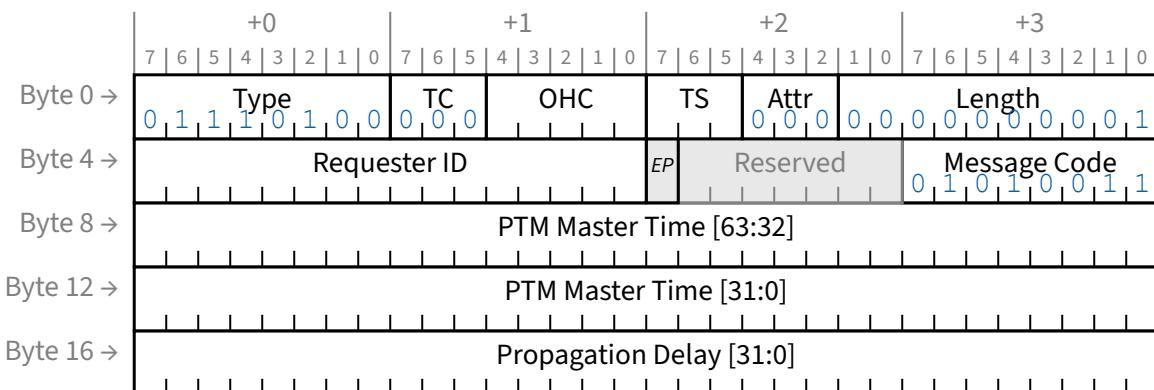


Figure 2-70 PTM ResponseD Message - Flit Mode §

IMPLEMENTATION NOTE: PROPAGATION DELAY[31:0] ENDIANNESSE §

The bytes within the Propagation Delay[31:0] field (shown in [↑↑↑ Figure 2-68↑](#)) are such that:

- Data Byte 0 contains Propagation Delay [31:24]
- Data Byte 1 contains Propagation Delay [23:16]
- Data Byte 2 contains Propagation Delay [15:8]
- Data Byte 3 contains Propagation Delay [7:0]

Due to ambiguity in previous versions of this document, some implementations made this interpretation:

- Data Byte 0 contains Propagation Delay [7:0]
- Data Byte 1 contains Propagation Delay [15:8]
- Data Byte 2 contains Propagation Delay [23:16]
- Data Byte 3 contains Propagation Delay [31:24]

As a result, it is recommended that implementations provide mechanisms for adapting to either byte interpretation. One such mechanism is the optional PTM Propagation Delay Adaptation Capability.

2.2.8.11 Integrity and Data Encryption (IDE) Messages §

IDE Messages are used with the optional Integrity and Data Encryption (IDE) mechanism (see [§ Section 6.33](#)). The following rules apply to the formation of IDE Messages:

- [§ Table 2-36](#) defines the IDE Messages.
- The IDE Messages do not include a data payload (TLP Type is Msg).
- The Length field is Reserved.

- The Requester ID must be set to the RID of the Function implementing IDE at the Transmitting Port.
 - IDE Sync and IDE Fail Messages associated with a Link IDE Stream must use Local **↑↓- Terminate at Receiver** routing (100b).
 - IDE Sync and IDE Fail Messages associated with a Selective IDE Stream must use **↑↓Route↓** **↑↓Routed↑** by ID (010b), and the Destination ID must contain the value in the RID Base field of the Selective IDE RID Association Register Block.

These Messages must only be Transmitted if the Valid bit is Set in the Selective IDE RID Association Register for the Selective IDE Stream.

- IDE Sync and IDE Fail Messages must use the same Traffic Class designator as the associated IDE Stream, if the Traffic Class designator maps to a non-UIO VC. IDE Fail and IDE Sync messages are not architected for Traffic Class designators that map to a UIO VC.
 - IDE Sync Messages are implicitly associated with the same IDE Stream as indicated in the IDE Prefix applied to the IDE Sync Message .

Table 2-36 IDE Messages

| Name | TLP Type | Code[7:0] (b) | Routing r[2:0] (b) | Support ¹ | | | | Description/Comments |
|----------|----------|---------------|--------------------|----------------------|----|----|----|---|
| | | | | RC | EP | Sw | Br | |
| IDE Sync | Msg | 0101 0100 | 010 / 100 | tr | tr | tr | | Synchronization of IDE PR Count for the associated IDE Stream |
| IDE Fail | Msg | 0101 0101 | 010 / 100 | tr | tr | tr | | Notification of IDE failure for a specific IDE Stream from the detecting Port to the IDE Partner Port |

Notes:

1. Support for these messages is required when the optional IDE mechanism is implemented

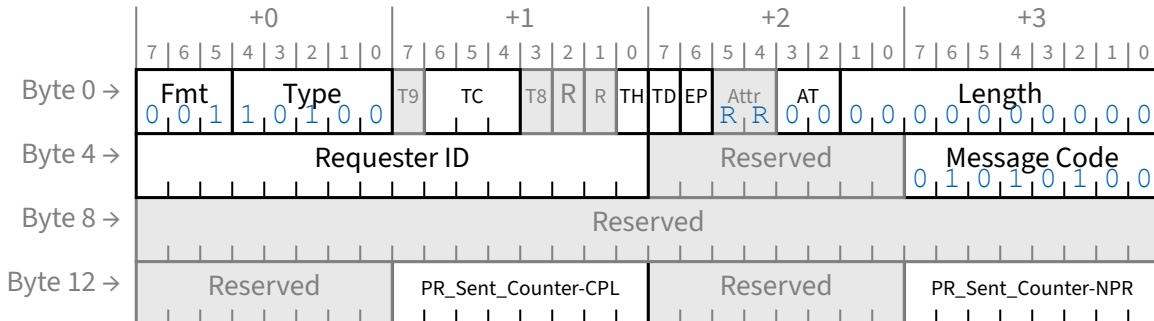


Figure 2-71 IDE Sync Message for Link IDE Stream - Non-Flit Mode §

Base 6.4 vs Base 6.3

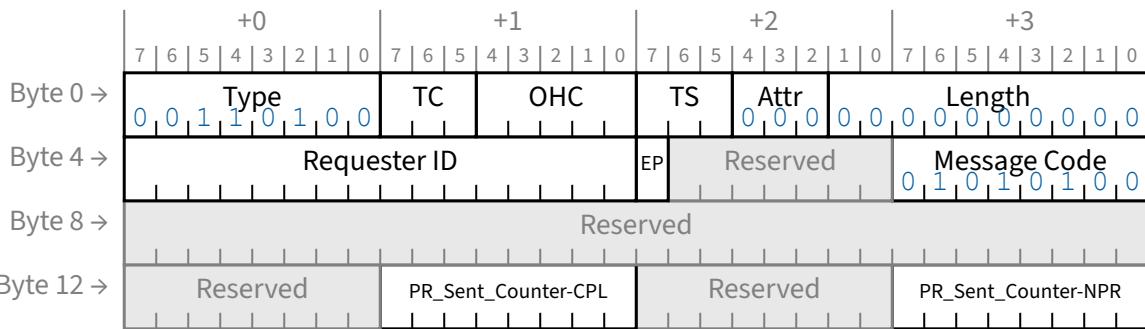


Figure 2-72 IDE Sync Message for Link IDE Stream - Flit Mode §

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [{"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": [0, 0, 1], "name": "Fmt", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "value": [1, 0, 0, 0, 1, 0, 0], "name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "value": [T9], "name": "T9", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "value": [TC], "name": "TC", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 5, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 4, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 3, "msbyte": 2, "msbit": 3, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 2, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 1, "msbyte": 2, "msbit": 1, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 0, "msbyte": 2, "msbit": 0, "value": [R], "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 7, "msbyte": 7, "msbit": 7, "value": [TH], "name": "TH", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 6, "msbyte": 7, "msbit": 6, "value": [TD], "name": "TD", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 5, "msbyte": 7, "msbit": 5, "value": [EP], "name": "EP", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 4, "msbyte": 7, "msbit": 4, "value": [Attr], "name": "Attr", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 3, "msbyte": 7, "msbit": 3, "value": [AT], "name": "AT", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 2, "msbyte": 7, "msbit": 2, "value": [Length], "name": "Length", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 1, "msbyte": 7, "msbit": 1, "value": [Message Code], "name": "Message Code", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 7, "msbit": 0, "value": [Destination RID], "name": "Destination RID", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 7, "msbyte": 7, "msbit": 7, "value": [Requester ID], "name": "Requester ID", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 6, "msbyte": 7, "msbit": 6, "value": [Reserved], "name": "Reserved", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 5, "msbyte": 7, "msbit": 5, "value": [Message Code], "name": "Message Code", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 4, "msbyte": 7, "msbit": 4, "value": [Reserved], "name": "Reserved", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 3, "msbyte": 7, "msbit": 3, "value": [Reserved], "name": "Reserved", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 2, "msbyte": 7, "msbit": 2, "value": [PR_Sent_Counter-CPL], "name": "PR_Sent_Counter-CPL", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 1, "msbyte": 7, "msbit": 1, "value": [PR_Sent_Counter-NPR], "name": "PR_Sent_Counter-NPR", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro"} ]} ↓
```

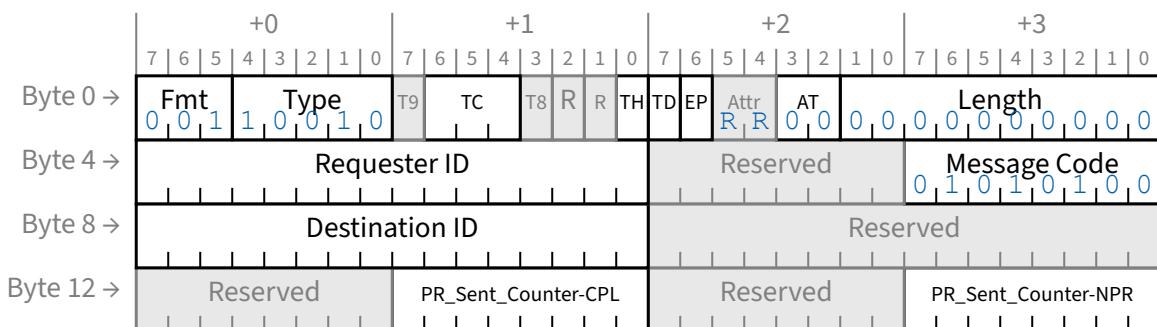


Figure 2-73 IDE Sync Message for Selective IDE Stream - Non-Flit Mode §

```
↑|↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "value": ["0", "0", "1", "1", "0", "0", "1", "0"], "attr": "ro" },
  { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "value": ["0", "0", "0", "0"], "attr": "ro" },
  { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro" },
  { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" },
  { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": ["0", "1", "0", "1", "0", "0", "1", "0"] },
  { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination RID", "attr": "ro" },
  { "lsbyte": 10, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 12, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 13, "lsbit": 0, "msbyte": 13, "msbit": 7, "name": "PR_Sent_Counter CPL", "addClass": "regFieldSmallText", "attr": "ro" },
  { "lsbyte": 14, "lsbit": 0, "msbyte": 14, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 7, "name": "PR_Sent_Counter NPR", "addClass": "regFieldSmallText", "attr": "ro" }
]}↓]
```

Base 6.4 vs Base 6.3

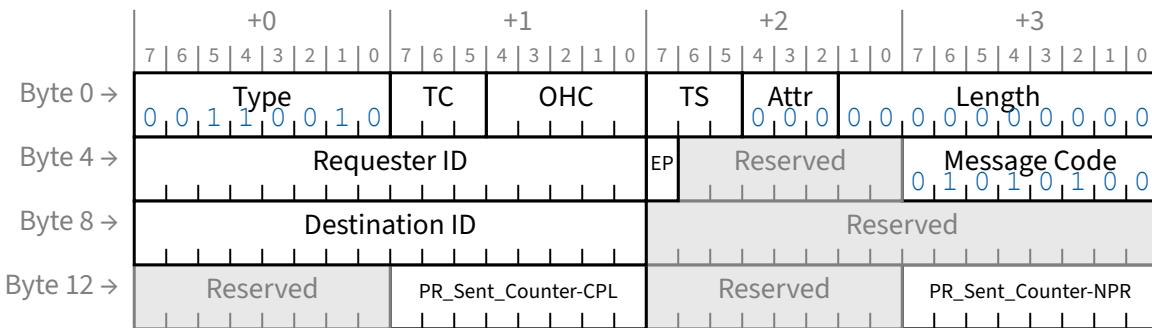


Figure 2-74 IDE Sync Message for Selective IDE Stream - Flit Mode §

Base 6.4 vs Base 6.3

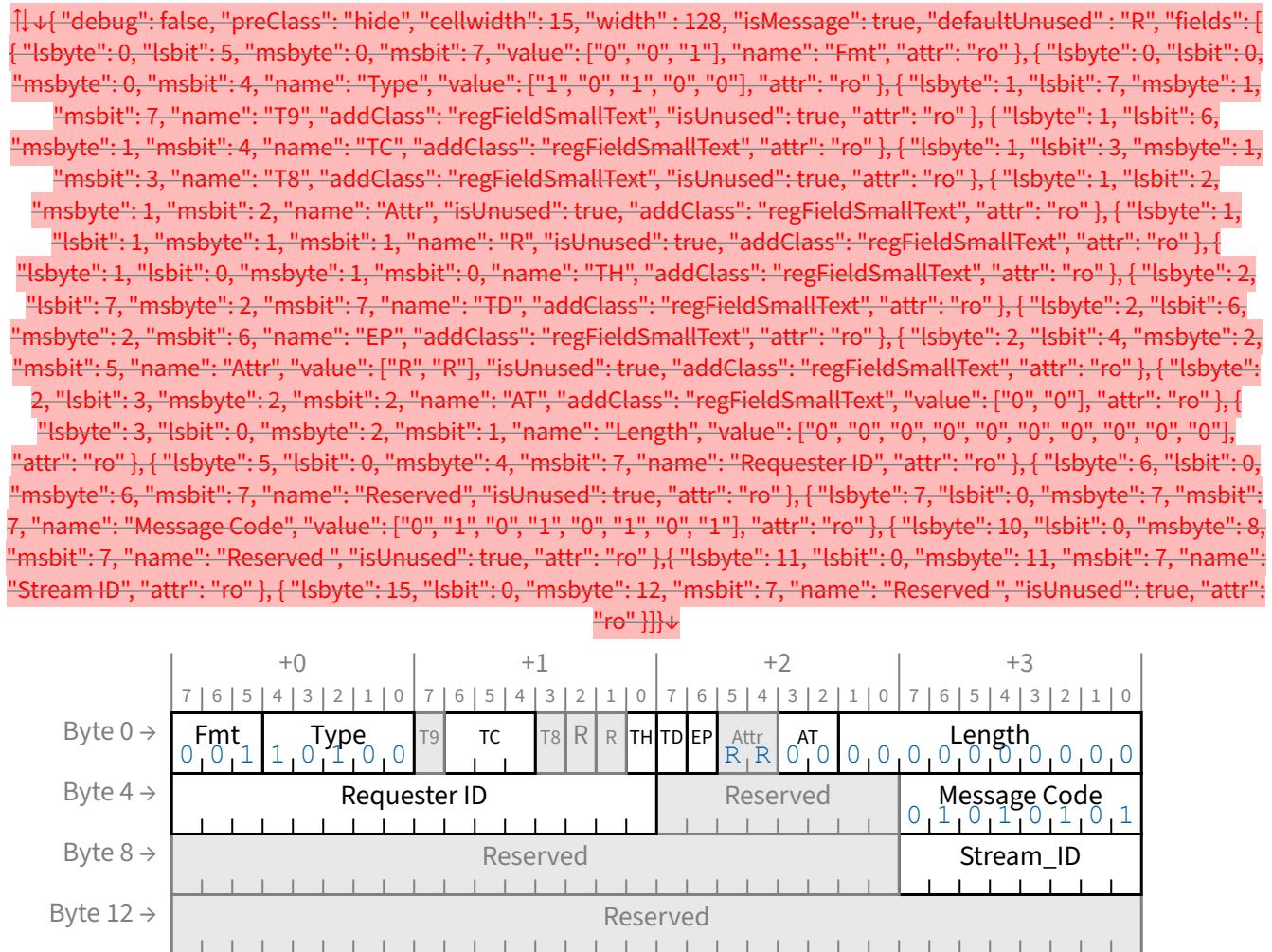


Figure 2-75 IDE Fail Message for Link IDE Stream - Non-Flit Mode §

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [{ "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "value": ["0", "0", "1", "1", "0", "1", "0", "0"], "name": "Type", "value": ["0", "0", "1", "1", "0", "1", "0", "0"], "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Attr", "value": ["0", "0", "0", "0"], "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 0, "msbit": 2, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "R", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 6, "msbit": 6, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Message Code", "value": ["0", "0", "1", "0", "1", "0", "0"], "attr": "ro" }, { "lsbyte": 10, "lsbit": 0, "msbyte": 8, "msbit": 7, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 7, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Stream ID", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "value": ["1", "0", "1", "0", "0", "0", "0", "0"], "name": "Reserved", "isUnused": true, "attr": "ro" }] }] } ↓↑

Base 6.4 vs Base 6.3

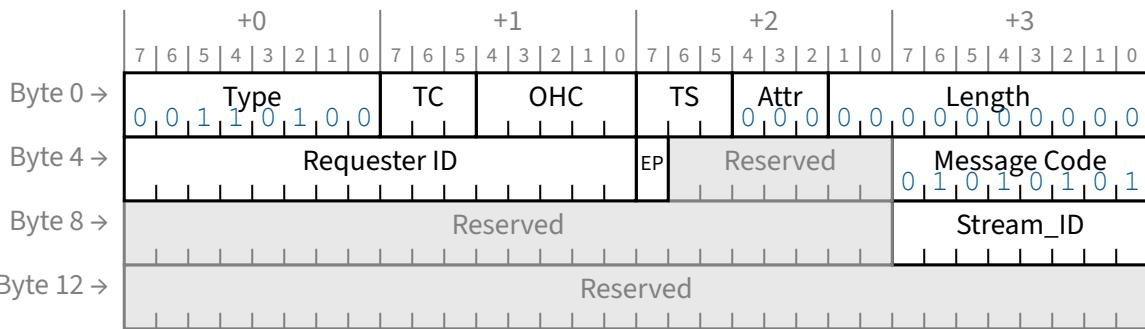


Figure 2-76 IDE Fail Message for Link IDE Stream - Flit Mode §

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": [ "0", "0", "1" ], "name": "Fmt", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "name": "Type", "value": [ "1", "0", "0", "1", "0", "0" ], "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8", "addClass": "regFieldSmallText", "isUnused": true, "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "Attr", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 1, "msbyte": 1, "msbit": 1, "name": "R", "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr", "value": [ "R", "R" ], "isUnused": true, "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 3, "msbyte": 2, "msbit": 2, "name": "AT", "addClass": "regFieldSmallText", "value": [ "0", "0" ], "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": [ "0", "0", "0", "0", "0", "0", "0", "0" ], "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" }, { "lsbyte": 6, "lsbit": 6, "msbyte": 6, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": [ "0", "1", "0", "1", "0", "1", "0" ], "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination RID", "attr": "ro" }, { "lsbyte": 10, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 7, "name": "Stream ID", "attr": "ro" }, { "lsbyte": 10, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 7, "name": "Stream ID", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" } ] } ↓↑
```

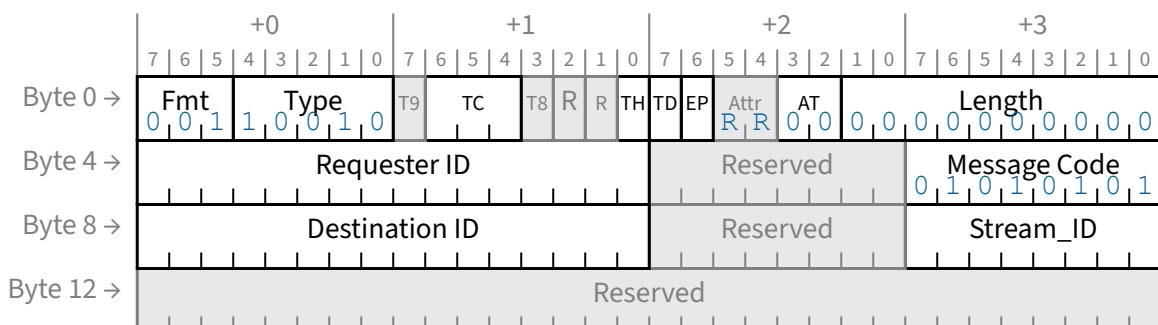


Figure 2-77 IDE Fail Message for Selective IDE Stream - Non-Flit Mode §

```
↑↓↔{ "debug": false, "preClass": "hide", "cellwidth": 15, "width" : 128, "isMessage": true, "defaultUnused": "R", "fields": [
  { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "value": ["0", "0", "1", "1", "0", "0", "1", "0"], "attr": "ro" },
  { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" },
  { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" },
  { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "value": ["0", "0", "0", "0"], "attr": "ro" },
  { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro" },
  { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "attr": "ro" },
  { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "EP", "addClass": "regFieldSmallText", "attr": "ro" },
  { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Message Code", "value": ["0", "1", "0", "1", "0", "1", "0", "1"] },
  { "lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination RID", "attr": "ro" },
  { "lsbyte": 10, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 7, "name": "Stream ID", "attr": "ro" },
  { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" },
  { "lsbyte": 15, "lsbit": 0, "msbyte": 12, "msbit": 7, "name": "Reserved", "isUnused": true, "attr": "ro" } ] }↓
```

Base 6.4 vs Base 6.3

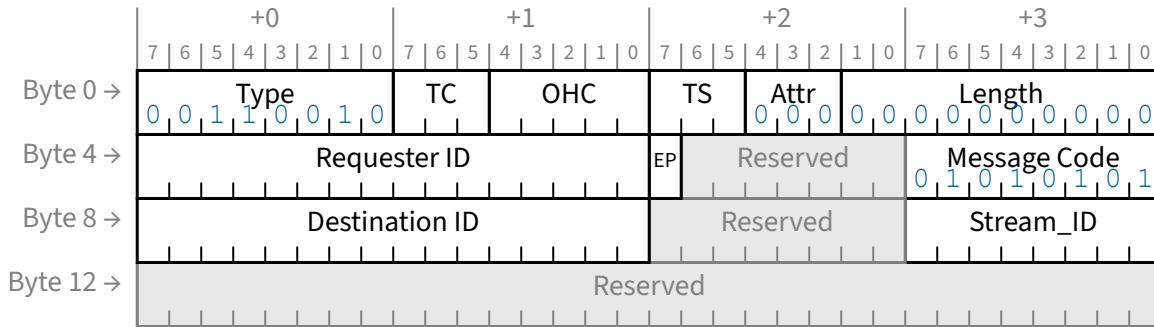


Figure 2-78 IDE Fail Message for Selective IDE Stream - Flit Mode §

2.2.9 Completion Rules §

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

2.2.9.1 Completion Rules for Non-Flit Mode §

- Completions route by ID, and use a 3 DW header.
 - Note that the routing ID fields correspond directly to the Requester ID supplied with the corresponding Request. Thus, for Completions these fields will be referred to collectively as the Requester ID instead of the distinct fields used generically for ID routing.
- In addition to the header fields included in all TLPs and the ID routing fields, Completions contain the following additional fields (see § Figure 2-79):
 - Completer ID[15:0] - Identifies the Completer - described in detail below
 - Completion Status[2:0] - Indicates the status for a Completion (see § Table 2-37)
 - Rules for determining the value in the Completion Status[2:0] field are in § Section 2.3.1
 $\uparrow\downarrow.\uparrow\uparrow\text{and } \S\text{ Section 10.2.3}\uparrow$

- BCM - Byte Count Modified - this bit must not be set by PCI Express Completers, and may only be set by PCI-X completers
- Byte $\text{↓Count}[11:0]$ ↑Count - The remaining Byte Count for Request
 - The Byte Count value is specified as a binary number, with 0000 0000 0001b indicating 1 byte, 1111 1111 1111b indicating 4095 bytes, and 0000 0000 0000b indicating 4096 bytes.
 - For Memory Read Completions, Byte $\text{↓Count}[11:0]$ ↑Count is set according to the rules in § Section 2.3.1.1.
 - For AtomicOp Completions, the Byte Count value must equal the associated AtomicOp operand size in bytes.
 - For all other types of Completions, the Byte Count value must be 4.
- Tag[9:0] - in combination with the Requester ID field, corresponds to the Transaction ID. In Non-Flit Mode, the Tag field is 10 bits.
- Lower Address[6:0] - lower byte address for starting byte of Completion
 - For Memory Read Completions, the value in this field is the byte address for the first enabled byte of data returned with the Completion (see the rules in § Section 2.3.1.1).
 - For AtomicOp Completions, the Lower Address field is Reserved.
 - This field is set to all 0's for all remaining types of Completions. Receivers may optionally check for violations of this rule. See § Section 2.3.2, second bullet, for details.

Base 6.4 vs Base 6.3

```

↓↓[{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [
    {"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "name": "Fmt", "value": ["0", "x", "0"], "attr": "ro"}, {"lsbyte": 0, "lsbit": 0,
        "msbyte": 0, "msbit": 4, "name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "T9",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 4, "name": "TC",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 3, "msbyte": 1, "msbit": 3, "name": "T8",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 2, "name": "A2",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 0, "name": "TH",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 7, "name": "TD",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 2, "msbit": 6, "name": "EP",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Attr",
        "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 3, "name": "AT",
        "addClass": "regFieldVerySmallText regFieldReservedText", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0,
        "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Completer ID",
        "attr": "ro"}, {"lsbyte": 6, "lsbit": 5, "msbyte": 6, "msbit": 7, "name": "Cpl. Status", "addClass": "regFieldVerySmallText",
        "attr": "ro"}, {"lsbyte": 6, "lsbit": 4, "msbyte": 6, "msbit": 4, "name": "BCM", "addClass": "regFieldVerySmallText regFieldReservedText",
        "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "Byte Count", "attr": "ro"}, {"lsbyte": 10, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "Tag", "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 6, "name": "Lower Address", "attr": "ro"}]]↓

```

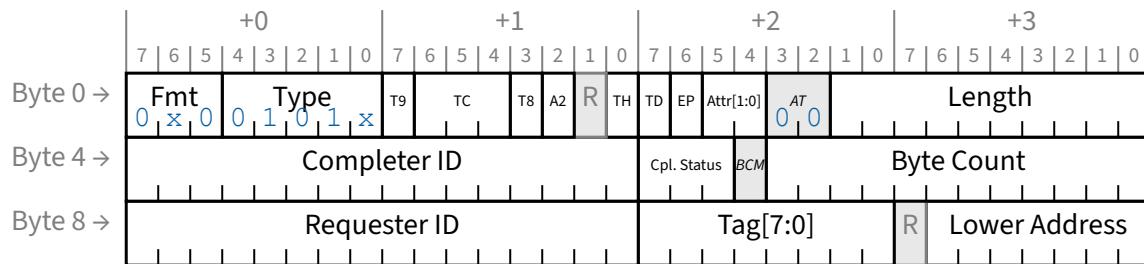


Figure 2-79 Completion Header Format - Non-Flit Mode §

Table 2-37 Completion Status Field Values §

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

- The Completer ID[15:0] is a 16-bit value that is unique for every PCI Express Function within a Hierarchy (see § Figure 2-80 and § Figure 2-81)

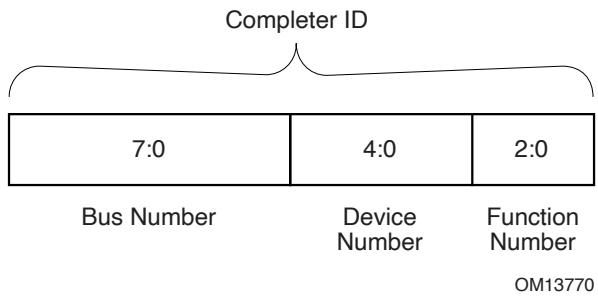


Figure 2-80 (Non-ARI) Completer ID §

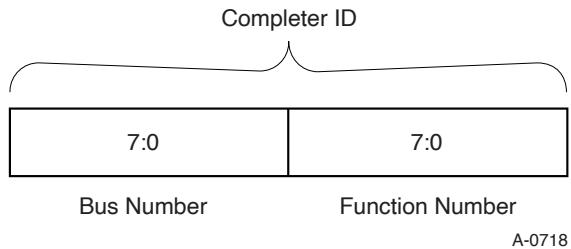


Figure 2-81 ARI Completer ID §

- Functions must capture the Bus and Device Numbers³³ supplied with all Type 0 Configuration Write Requests completed by the Function, and supply these numbers in the Bus and Device Number fields of the Completer ID³⁴ for all Completions generated by the Device/Function.
 - If a Function must generate a Completion prior to the initial device Configuration Write Request, 0's must be entered into the Bus Number and Device Number fields
 - Note that Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Write Request.
 - Exception: The assignment of Bus Numbers to the Devices within a Root Complex may be done in an implementation specific way.
- In some cases, a Completion with UR Completion Status may be generated by an MFD without associating the Completion with a specific Function within the device - in this case, the Function Number field³⁵ is Reserved.
 - Example: An MFD receives a Read Request that does not target any resource associated with any of the Functions of the device - the device generates a Completion with UR status and sets a value of all 0's in the Function Number field of the Completer ID.
- Completion headers must supply the same values for the Requester ID, Tag, and Traffic Class as were supplied in the header of the corresponding Request.
- Completion headers must supply the same values for the Attribute as were supplied in the header of the corresponding Request, except as explicitly allowed:
 - when IDO is used (see § Section 2.2.6.4)

33. With ARI Devices, Functions are only required to capture the Bus Number. ARI Devices are permitted to retain the captured Bus Number on either a per-Device or a per-Function basis. See § Section 2.2.6.2 .

34. An ARI Completer ID does not contain a Device Number field. See § Section 2.2.4.2 .

35. With an ARI Completer ID, the Function Number field is 8 bits.

- when RO is used in a Translation Completion (see § Section 10.2.3)
- ~~The TH bit is reserved for Completions.~~ Reserved.
- AT[1:0] must be 00b. Receivers are not required or encouraged to check this.
- The Completer ID field is not meaningful prior to the software initialization and configuration of the completing device (using at least one Configuration Write Request), and for this case the Requester must ignore the value returned in the Completer ID field.
- A Completion including a data payload must specify the actual amount of data returned in that Completion, and must include the amount of data specified.
 - It is a TLP formation error to include more or less data than specified in the Length field, and the resulting TLP is a Malformed TLP.

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

2.2.9.2 Completion Rules for Flit Mode §

In Flit Mode, the rules for non-UIO Completions are the same as in Non-Flit Mode, except as defined in this section. In Flit Mode, non-UIO Completions must use the Completion Header Base Format shown in § Figure 2-82 . UIO Write Completions and UIO Read Completions with Completion Status other than Successful Completion (i.e., without Data) must use the ~~Completion~~ Header Base Format shown in § Figure 2-83 . UIO Read Completions with Data must use the UIO Completion Header Base Format shown in § Figure 2-84 .

In Flit Mode, the Tag field is 14 bits.

~~In Flit Mode, Lower Address[6], Lower Address[5:2], and Lower Address[1:0], are not contiguous field bits in the TLP.
In § Figure 2-82 to § Figure 2-84 , LA6 is Lower Address[6] and LA[5:2] is Lower Address[5:2].~~

Reserved Completions (as indicated in § Table 2-5), are ID Routed TLPs as defined in § Section 2.2.4.2 .

Base 6.4 vs Base 6.3

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Completer ID", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "LA[6]", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Tag[13:0]", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "LA[5:2]", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 10, "lsbit": 3, "msbyte": 11, "msbit": 0, "name": "Byte Count[11:0]", "addClass": "regFieldSmallText", "attr": "ro" } ] }↑
```

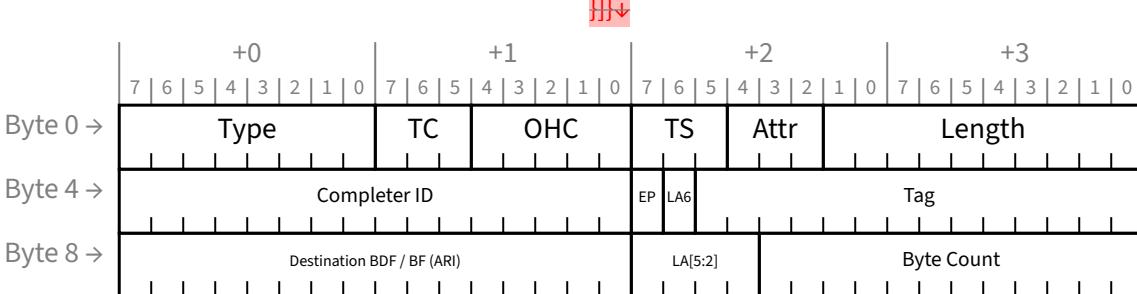


Figure 2-82 Completion Header Base Format - Non-UIO Flit Mode §

```
↑↓{ "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro" }, { "lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Completer ID", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 6, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "LA[6]", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Tag[13:0]", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "LA[5:2]", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 11, "lsbit": 6, "msbyte": 11, "msbit": 5, "name": "CDL[1:0]", "addClass": "regFieldVerySmallText", "attr": "ro" } ] }↑
```

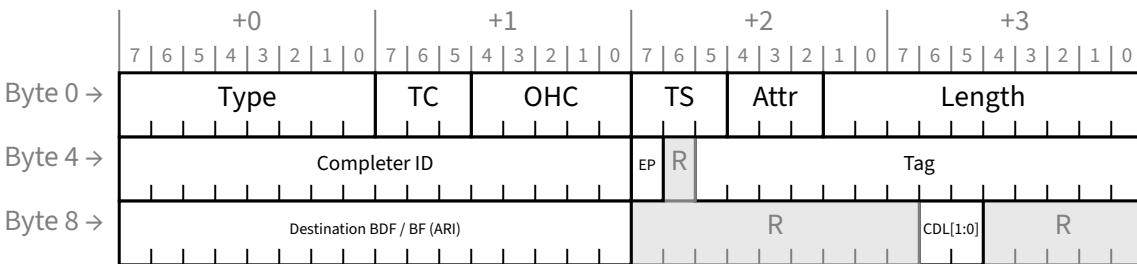


Figure 2-83 Completion Header Base Format – UIOWrCpl and UIORDCpl Flit Modes §

Base 6.4 vs Base 6.3

```
↓↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [{"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Type", "attr": "ro"}, {"lsbyte": 1, "lsbit": 5, "msbyte": 1, "msbit": 7, "name": "TC", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 4, "name": "OHC", "attr": "ro"}, {"lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "TS", "attr": "ro"}, {"lsbyte": 2, "lsbit": 2, "msbyte": 2, "msbit": 4, "name": "Attr", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Completer ID", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "EP", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 6, "lsbit": 6, "msbyte": 6, "msbit": 6, "name": "LA[6]", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag[13:0]", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 9, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Destination BDF / BF (ARI)", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 10, "lsbit": 4, "msbyte": 10, "msbit": 7, "name": "LA[5:2]", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 6, "msbyte": 5, "name": "CDL[1:0]", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 4, "msbyte": 11, "msbit": 0, "name": "LA[11:7]", "addClass": "regFieldSmallText", "attr": "ro"}]}↓
```

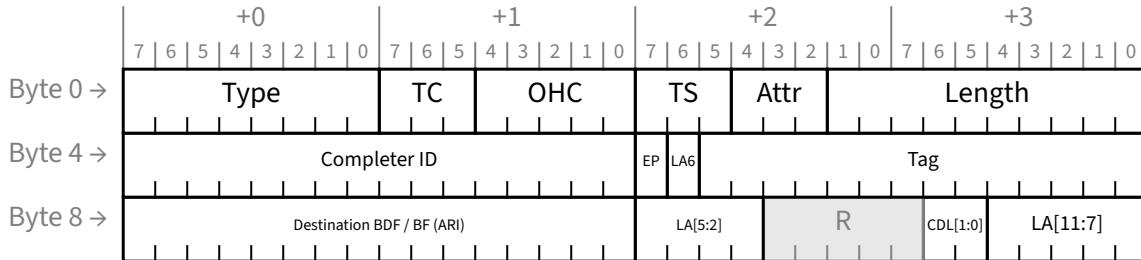


Figure 2-84 Completion Header Base Format - UIORdCplID ↓↑- Flit Mode §

OHC-A5 (see § Figure 2-11) is required for all:

- Unsuccessful Completions
- Non-UIO Completions with Lower Address[1:0] not equal to 00b
- Completions that require the Destination Segment due to the associated Non-Posted Request ↓↑or UIO Request containing a Requester Segment that does not match the Completer's captured Segment.

Errata: Base 6.3
B834△◀▶

When OHC-A5 is not present it is implied that the Completion Status is ↓↓Successful, ↓↓Successful Completion, that Completer Segment and Destination Segment need not be explicitly indicated (see Segment rules in § Section 2.2.1.2), and that, for non-UIO Completions, the Lower Address[1:0] = 00b.

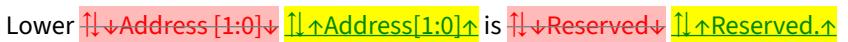
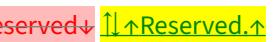
When OHC-A5 is present:

- ↓↑The Completion Status and, for non-UIO Completions, Lower Address[1:0] fields must contain valid ↓↓values ↓↓values.
 - For UIO Completions, Lower Address[1:0] is ↓↓Reserved ↓↓Reserved.
- ↓↑If the Segment Captured bit is Set, the Completer Segment field must contain the Segment value captured by the Function as described in § Section 2.2.6.2; ↓↑if the Segment Captured bit is Clear, the Completer Segment field must be ↓↓00h if Segment Captured is Clear ↓↓00h.
- if the associated Request did not include a Requester Segment, the Destination Segment field must be 00h and the DSV bit must be ↓↓clear. ↓↓Clear. If the associated Request included a Requester Segment, the

Destination Segment field must reflect the value of the Requester Segment and the DSV bit must be Set. See RP Segment Exceptions for cases where RPs are not required to include Segment information.

The BCM field, present in Non-Flit Mode Completions, is not supported in Flit Mode.

For all UIO Completions:

- The Read Completion Boundary and Write Completion Boundary rules defined in § Section 2.3.1.2 and § Section 2.3.1.3, respectively, must be followed.
- Length[9:0] indicates the total number of DW represented by this Completion. See § Table 2-4 for values.
 - Regardless of Completion Status, Completers must return Completions corresponding to all DW in a UIO Request.
 - Byte Enables must not be considered when determining the Length value for UIO Completions.
 - For a Zero Length UIO Write (where in the Request, Length is 00_0000_0001b and First Byte Enable 0000b), one DW must be considered to have been written.
- The Tag field value must match the Tag field value for the corresponding UIO Request(s)
 - UIO Write Completions are permitted to be coalesced or split, for a given Transaction ID, provided all DW Completion accounting remains accurate (see § Section 2.3.1.3).
 - UIO Read Completions are permitted to be split, for a given Transaction ID, provided all DW Completion accounting remains accurate (see § Section 2.3.1.2).
- For UIO Completions without Data (see § Figure 2-83)
 - Lower Address  is Reserved 
- For UIO Completions with Data (see § Figure 2-84)
 - Lower  must contain valid 
 - Lower  is 
- The CDL[1:0] field is assigned for use by [CXL]; this field must be treated as Reserved for use cases not covered by [CXL].
- UIO Requesters must accept UIO Completions in any order.
- UIO Memory Request(s) associated with a Transaction ID are considered completed only when the sum of all DW completed, as indicated by the Length[9:0] field value(s) in the Completion(s), equals the sum of all DW expected for the associated Request(s).
 - Only UIO Memory Writes are allowed to have multiple outstanding Requests with the same Transaction ID. If it is necessary for a Requester to ensure that UIO Memory Write Requests issued with a given Transaction ID have completed, the Requester must delay issuing additional Requests with that Transaction ID until it has received Completions accounting for all outstanding UIO Memory Write Requests using that Transaction ID.
- When fully completed, all Requests associated with the same Transaction ID are represented by the same Completion Status. However, individual Completions of a UIO Request may indicate different Completion Status values. At any point where UIO Completions are coalesced, including at the Requester, the coalesced Completion Status is determined according to the following rules:
 - UR if any of the UIO Completions have UR Completion Status
 - CA if none of the UIO Completions have UR Completions Status, and any have CA Completion Status
 - RRS if none of the UIO Completions have UR or CA Completions Status, and any have RRS Completion Status
 - SC if all UIO Completions have Successful Completion Status

- Any completion status not defined in § Table 2-37 should be treated as a UR (see § Section 2.3.2)
- UIO Completions for UIO Read Requests that have a Completion Status other than Successful Completion must use TLP Type UIORdCpl
 - The Length value for UIORdCpl is not constrained by Max Payload Size or Max Read Request Size.
- Attr[2:0], corresponding to IDO, RO and NS in non-UIO Memory Requests, are Reserved
- The EP 11↑bit1 is Reserved for UIOWrCpl and UIORdCpl.

2.2.10 TLP Prefix Rules §

2.2.10.1 TLP Prefix General Rules - Non-Flit Mode §

In NFM, the following rules apply to any TLP that contains a TLP Prefix:

- For any TLP, a value of 100b in the Fmt[2:0] field in byte 0 of the TLP indicates the presence of a TLP Prefix and the Type[4] bit indicates the type of TLP Prefix.
 - A value of 0b in the Type[4] bit indicates the presence of a Local TLP Prefix
 - A value of 1b in the Type[4] bit indicates the presence of an End-End TLP Prefix
- The format for bytes 1 through 3 of a TLP Prefix is defined by its TLP Prefix type.
- A TLP that contains a TLP Prefix must have an underlying TLP Header. A received TLP that violates this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (see § Section 6.2).
- It is permitted for a TLP to contain more than one TLP Prefix of any type
 - When a combination of Local and End-End TLP Prefixes are present in TLP, it is required that all the Local TLP Prefixes precede any End-End TLP Prefixes. A received TLP that violates this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (see § Section 6.2).
- The size of each TLP Prefix is 1 DW. A TLP Prefix may be repeated to provide space for additional data.
- If the value in the Fmt and Type field indicates the presence of a Local TLP Prefix, handle according to the Local TLP Prefix handling (see § Section 2.2.10.2).
- If the value in the Fmt and Type field indicates the presence of an End-End TLP Prefix, handle according to the End-End TLP Prefix handling (see § Section 2.2.10.4).

2.2.10.2 Local TLP Prefix Processing §

The following rules apply to Local TLP Prefixes:

- In Flit Mode, TLP Prefix types are determined using the Type[7:0] field (see § Table 2-5)
- In Non-Flit Mode, Local TLP Prefix types are determined using the L[3:0] sub-field of the Type field
 - Type[4] must be 0b

Local TLP Prefix L[3:0] values are defined in § Table 2-38 11↑↑

Table 2-38 Local TLP Prefix Types §

| Local TLP Prefix Type | L[3:0] (b) | Description |
|-----------------------|------------|---|
| MR-IOV | 0000 | MR-IOV TLP Prefix - Refer to [MR-IOV] for details. |
| FlitModePrefix | 1101 | Flit Mode Local TLP Prefix - See § Section 2.2.10.3 |

| Local TLP Prefix Type | L[3:0] (b) | Description |
|-----------------------|------------|---|
| VendPrefixL0 | 1110 | Vendor Defined Local TLP Prefix - Refer to § Section 2.2.10.2.1 for further details. |
| VendPrefixL1 | 1111 | Vendor Defined Local TLP Prefix - Refer to § Section 2.2.10.2.1 for further details. |
| | | All other encodings are Reserved. |

- The size, routing, and flow control rules are specific to each Local TLP Prefix type.
- It is an error to receive a TLP with a Local TLP Prefix type not supported by the Receiver. If the Extended Fmt Field Supported bit is Set, TLPs in violation of this rule are handled as a Malformed TLP unless explicitly stated differently in another specification. This is a reported error associated with the Receiving Port (see § Section 6.2). If the Extended Fmt Field Supported bit is Clear, behavior is device specific.
- No Local TLP Prefixes are protected by ECRC even if the underlying TLP is protected by ECRC.

2.2.10.2.1 Vendor Defined Local TLP Prefix §

As described in § Table 2-38 , Types VendPrefixL0 and VendPrefixL1 are defined for use as Vendor Defined Local TLP Prefixes. To maximize interoperability and flexibility the following rules are applied to such prefixes:

- Components must not send TLPs containing Vendor Defined Local TLP Prefixes unless this has been explicitly enabled (using vendor-specific mechanisms).
- Components that support any usage of Vendor Defined Local TLP Prefixes must support the 3-bit definition of the Fmt field and have the Extended Fmt Field Supported bit Set (see § Section 7.5.3.15).
- It is recommended that components be configurable (using vendor-specific mechanisms) so that all vendor defined prefixes can be sent using either of the two Vendor Defined Local TLP Prefix encodings. Such configuration need not be symmetric (for example each end of a Link could transmit the same Prefix using a different encoding).

2.2.10.3 Flit Mode Local TLP Prefix §

This prefix (see § Figure 2-85) is only permitted when operating in Flit Mode.

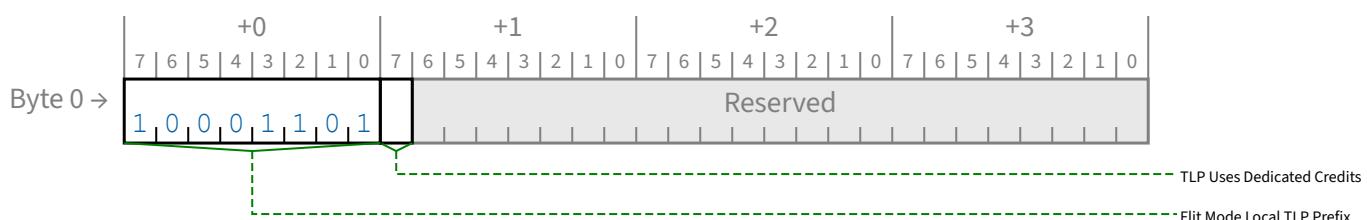


Figure 2-85 Flit Mode Local TLP Prefix §

If the Flit Mode Local TLP Prefix is applied to a NFM TLP, this is an error that *MUST@FLIT* be handled as a Malformed TLP. It is permitted to apply the Flit Mode Local TLP Prefix to any FM TLP, but it is strongly recommended that the Flit Mode Local TLP Prefix is only applied to TLPs that specifically require the Prefix to be present.

The Flit Mode Local TLP Prefix includes:

- **TLP Uses Dedicated Credits** – This bit when Set indicates that the associated TLP must be handled using dedicated flow control credits. If this bit is Clear, or if the Flit Mode Local TLP Prefix is not present, the associated TLP must be handled using shared flow control credits.

2.2.10.4 End-End TLP Prefix Processing - Non-Flit Mode §

The following rules apply to End-End TLP Prefixes

- End-End TLP Prefix types are determined using the E[3:0] sub-field of the Type field
 - Type[4] must be 1b
 - End-End TLP Prefix E[3:0] values are defined in § Table 2-39

Table 2-39 End-End TLP Prefix Types §

| End-End TLP Prefix Type | E[3:0] (b) | Description |
|-------------------------|------------|--|
| TPH | 0000 | TPH - Refer to <u>§ Section 2.2.7.1.1</u> and § <u>Section 6.17</u> for further details. |
| PASID | 0001 | PASID - Refer to <u>§ Section 6.20.2.1</u> for further details. |
| IDE | 0010 | Identifies an IDE TLP - Refer to § <u>Section 6.33</u> for further details. |
| VendPrefixE0 | 1110 | Vendor Defined End-End TLP Prefix - Refer to § <u>Section 2.2.10.4.1</u> for further details. |
| VendPrefixE1 | 1111 | Vendor Defined End-End TLP Prefix - Refer to § <u>Section 2.2.10.4.1</u> for further details. |
| | | All other encodings are Reserved. |

- The maximum number of End-End TLP Prefixes permitted in a TLP is 4:
 - A Receiver supporting TLP Prefixes must check this rule. If a Receiver determines that a TLP violates this rule, the TLP is a Malformed TLP. This is a reported error associated with the Receiving Port (see § Section 6.2).
 - The presence of an End-End TLP Prefix does not alter the routing of a TLP. TLPs are routed based on the routing rules covered in § Section 2.2.4.
 - Functions indicate how many End-End TLP Prefixes they support by the Max End-End TLP Prefixes field in the Device Capabilities 2 register (see § Section 7.5.3.15).
 - For Root Ports, the Max End-End TLP Prefixes field is permitted to return a value indicating support for fewer End-End TLP Prefixes than what the Root Port hardware actually implements; however, the error handling semantics must still be based on the value contained in the field. TLPs received that contain more End-End TLP Prefixes than are supported by the Root Port must be handled as follows. It is recommended that Requests be handled as Unsupported Requests, but otherwise they must be handled as Malformed TLPs. It is recommended that Completions be handled as Unexpected Completions, but otherwise they must be handled as Malformed TLPs. For TLPs received by the Ingress Port, this is a reported error associated with the Ingress Port. For TLPs received internally to be transmitted out the Egress Port, this is a reported error associated with the Egress Port. See § Section 6.2.
 - For all other Function types, TLPs received that contain more End-End TLP Prefixes than are supported by a Function must be handled as Malformed TLPs. This is a reported error associated with the Receiving Port (see § Section 6.2).
- Advanced Error Reporting (AER) logging (if supported) occurs as specified in § Section 6.2.4.4.

- Switches must support forwarding of TLPs with up to 4 End-End TLP Prefixes if the ~~↓↑End-End TLP Prefix Supported~~ ↓↑End-End TLP Prefix Supported bit is Set.
- Different Root Ports with the ~~↓↑End-End TLP Prefix Supported~~ ↓↑End-End TLP Prefix Supported bit Set are permitted to report different values for Max End-End TLP Prefixes.
- All End-End TLP Prefixes are protected by ECRC if the underlying TLP is protected by ECRC.
- It is an error to receive a TLP with an End-End TLP Prefix by a Receiver that does not support End-End TLP Prefixes. A TLP in violation of this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (see § Section 6.2).
- Software should ensure that TLPs containing End-End TLP Prefixes are not sent to components that do not support them. Components where the Extended Fmt Field Supported bit is Clear may misinterpret TLPs containing TLP Prefixes.
- If one Function of an Upstream Port has the ~~↓↑End-End TLP Prefix Supported~~ ↓↑End-End TLP Prefix Supported bit Set, all Functions of that Upstream Port must handle the receipt of a Request addressed to them that contains an unsupported End-End TLP Prefix type as an Unsupported Request. This is a reported error associated with the Receiving Port (see § Section 6.2).
- If one Function of an Upstream Port has the ~~↓↑End-End TLP Prefix Supported~~ ↓↑End-End TLP Prefix Supported bit Set, all Functions of that Upstream Port must handle the receipt of a Completion addressed to them that contains an unsupported End-End TLP Prefix type as an Unexpected Completion. This is a reported error associated with the Receiving Port (see § Section 6.2).
- For Routing Elements, the End-End TLP Prefix Blocking bit in each Egress Port determines whether TLPs containing End-End TLP Prefixes can be transmitted via that Egress Port (see § Section 7.5.3.16). If forwarding is blocked the entire TLP is dropped and a TLP Prefix Blocked Error is reported. If the blocked TLP is a Non-Posted Request, the Egress Port returns a Completion with Unsupported Request Completion Status. The TLP Prefix Blocked Error is a reported error associated with the Egress Port (see § Section 6.2).
- For routing elements where Multicast is enabled (see § Section 6.14). End-End TLP Prefixes are replicated in all Multicast copies of a TLP. TLP Prefix Egress Blocking of Multicast packets is performed independently at each Egress Port.

2.2.10.4.1 Vendor Defined End-End TLP Prefix §

As described in § Table 2-39 , Types VendPrefixE0 and VendPrefixE1 are defined for use as Vendor Defined End-End TLP Prefixes. To maximize interoperability and flexibility the following rules are applied to such prefixes:

- Components must not send TLPs containing Vendor Defined End-End TLP Prefixes unless this has been explicitly enabled (using vendor-specific mechanisms).
- It is recommended that components be configurable (using vendor-specific mechanisms) to use either of the two Vendor Defined End-End TLP Prefix encodings. Doing so allows two different Vendor Defined End-End TLP Prefixes to be in use simultaneously within a single PCI Express topology while not requiring that every source understand the ultimate destination of every TLP it sends.

2.2.10.4.2 Root Ports with ~~↓↑End-End TLP Prefix Supported~~ ↓↑End-End TLP Prefix Supported §

Support for peer-to-peer routing of TLPs containing End-End TLP Prefixes between Root Ports is optional and implementation dependent. If an RC supports End-End TLP Prefix routing capability between two or more Root Ports, it must indicate that capability in each associated Root Port via the ~~↓↑End-End TLP Prefix Supported~~ ↓↑End-End TLP Prefix Supported bit in the Device Capabilities 2 register .

An RC is not required to support End-End TLP Prefix routing between all pairs of Root Ports that have the ~~↓↑End-End TLP Prefix Supported~~ ↑↓End-End TLP Prefix Supported bit Set. A Request with End-End TLP Prefixes that would require routing between unsupported pairs of Root Ports must be handled as a UR. A Completion with End-End TLP Prefixes that would require routing between unsupported pairs of Root Ports must be handled as an Unexpected Completion (UC). In both cases, this error is reported by the “sending” Port.

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

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

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

2.2.11 OHC-E Rules - Flit Mode §

End-End TLP Prefixes in Non-Flit Mode are replaced by [OHC-E](#) in Flit Mode (see § [Figure 2-86](#) , § [Figure 2-87](#) , and § [Figure 2-88](#)).

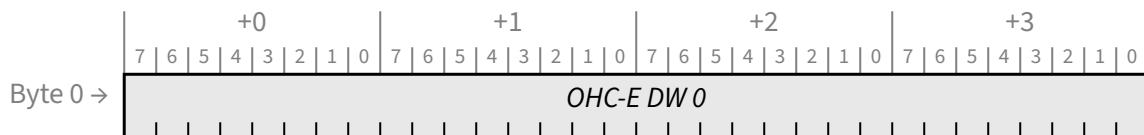


Figure 2-86 OHC-E1 §

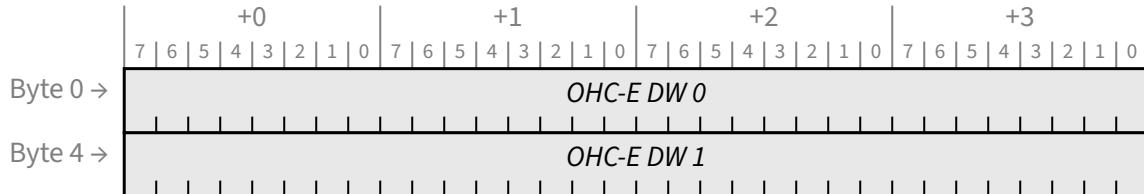


Figure 2-87 OHC-E2 §

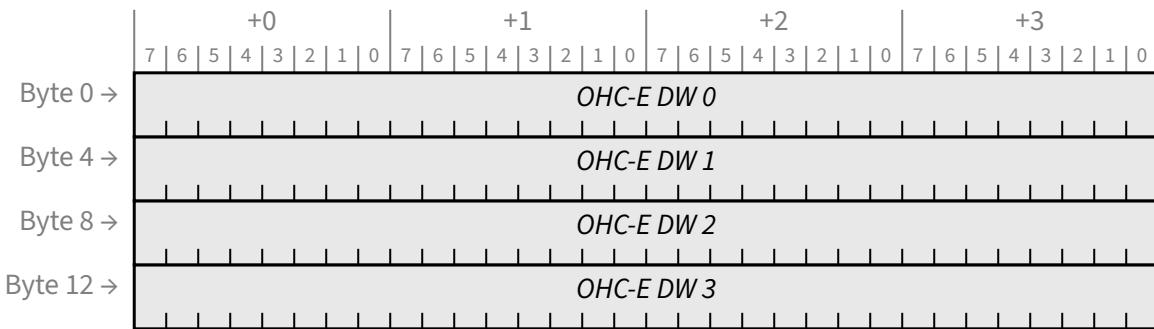


Figure 2-88 OHC-E §

OHC-E is used to convey content that would otherwise use End-End TLP Prefixes $\uparrow\downarrow 0011\downarrow$ $\uparrow\downarrow 0011b\uparrow$ to $\uparrow\downarrow 1111\downarrow$ $\uparrow\downarrow 1111b\uparrow$

- For each DW of OHC-E, Byte 0, bits [7:4] indicate the format of the remainder of the DW and are encoded:
 - 0000b - No Entry - The remainder of the DW is Reserved
 - 0001b - End-End TLP Prefix DW - The remainder of the DW is defined as follows:
 - Byte 0, bits [3:0] take the value of E[3:0] in the corresponding End-End TLP Prefix (see § Table 2-39), with the exception that encodings 0000b-0010b are Reserved.
 - Bytes 1, 2 and 3 take the value of bytes 1, 2 and 3 in the corresponding End-End TLP Prefix.
 - 0010b-1111b - Reserved - Receivers must handle as No Entry

OHC-E must be populated without gaps, starting with the first DW. Any No Entry DWs must be populated at the end. Transmitters must use the smallest possible OHC-E and avoid unnecessary No Entry DWs. When translating VendPrefixE0 or VendPrefixE1 from NFM to FM or vice-versa, the same relative sequence must be preserved.

RC support for peer-to-peer routing of TLPs containing OHC-E content between Root Ports is optional and implementation dependent.

If a Function sets does not support OHC-E, and it is the targeted Completer for a received TLP that has OHC-E, it must handle the TLP as an Unsupported Request or $\uparrow\downarrow an\uparrow$ Unexpected Completion. $\uparrow\downarrow The\uparrow$ $\uparrow\downarrow Such a\uparrow$ Function is permitted to drop OHC-E content during header logging for the error. This behavior is consistent with the rules stated in § Section 2.2.1.2 for Endpoint Upstream Ports and Root Ports, but with $\uparrow\downarrow an\uparrow$ extension of the rule for $\uparrow\downarrow switch ports\downarrow$ $\uparrow\downarrow Switch Ports\uparrow$ as well when they don't support OHC-E.

If a Switch Function or RP Function does not support OHC-E and it receives a TLP with OHC-E to be forwarded, the TLP must be handled as below.

- PR FC Type: Block at Ingress; if TLP is UIO, report no error, else handle as a TLP Prefix Blocked Error
- NPR FC Type: Block at Ingress; report no error
- CPL FC Type: Block at Ingress; handle as a TLP Prefix Blocked Error

If a Function sets its OHC-E Support field to 001b or 010b, it must handle a received TLP that targets it and that has OHC-E containing more DWs than it supports as an Unsupported Request or an Unexpected Completion.

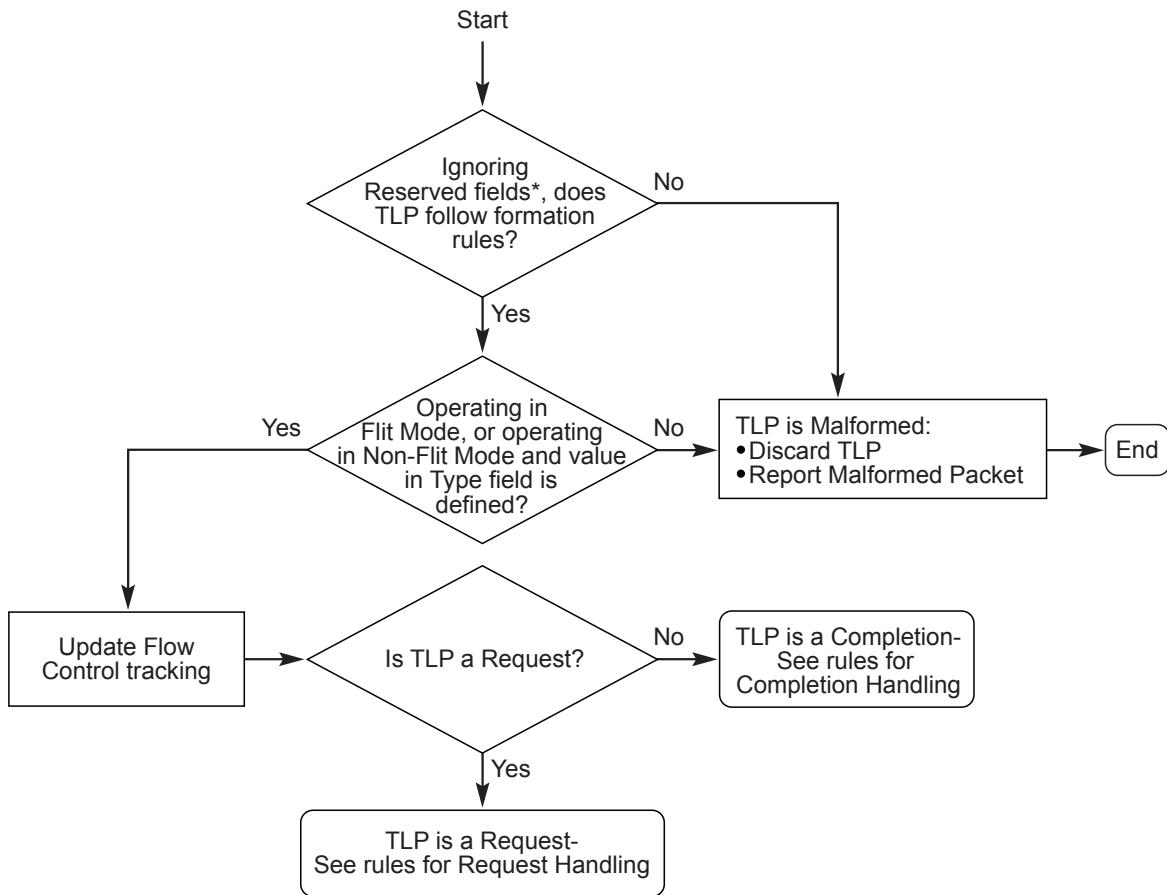
2.3 Handling of Received TLPs §

This section describes how all Received TLPs are handled when they are delivered to the Receive Transaction Layer from the Receive Data Link Layer, after the Data Link Layer has validated the integrity of the received TLP. The rules are diagrammed in the flowchart shown in § Figure 2-89.

- Values in Reserved fields must be ignored by the Receiver.
- In Non-Flit Mode, if the value in the Fmt field indicates the presence of at least one TLP Prefix:
 - Detect if additional TLP Prefixes are present in the header by checking the Fmt field in the first byte of subsequent DWs until the Fmt field does not match that of a TLP Prefix.
 - Handle all received TLP Prefixes according to TLP Prefix Handling Rules (see § [Section 2.2.10.1](#)).
- In Flit Mode, if the value in the Type field indicates the presence of at least one Local TLP Prefix:
 - Detect if additional Local TLP Prefixes are present in subsequent DWs.
 - Handle all received Local TLP Prefixes according to TLP Prefix Handling Rules (see § [Section 2.2.10.3](#)).
- In Non-Flit Mode, if the Extended Fmt Field Supported bit is Set, Received TLPs that use encodings of Fmt and Type that are Reserved are Malformed TLPs (see [§ Table 2-2](#) and [§ Table 2-3](#)).
 - This is a reported error associated with the Receiving Port (see § [Section 6.2](#)).
- In Non-Flit Mode, if the Extended Fmt Field Supported bit is Clear, processing of Received TLPs that have Fmt[2] Set is undefined.³⁶
 - All Received Malformed TLPs with Fmt[2] Clear and that use undefined Type field values are Malformed TLPs.
 - This is a reported error associated with the Receiving Port (see § [Section 6.2](#)).
 - All Received Malformed TLPs must be discarded.
 - Received Malformed TLPs that are ambiguous with respect to which buffer to release or are mapped to an uninitialized or disabled Virtual Channel must be discarded without updating Receiver Flow Control information.
 - All other Received Malformed TLPs must be discarded, optionally not updating Receiver Flow Control information.
- Otherwise, update Receiver Flow Control tracking information (see § [Section 2.6](#)).
- 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 so handle according to Completion Handling Rules (see § [Section 2.3.2](#)).

³⁶ An earlier version of this specification reserved the bit now defined for Fmt[2].

Base 6.4 vs Base 6.3



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

OM13771C

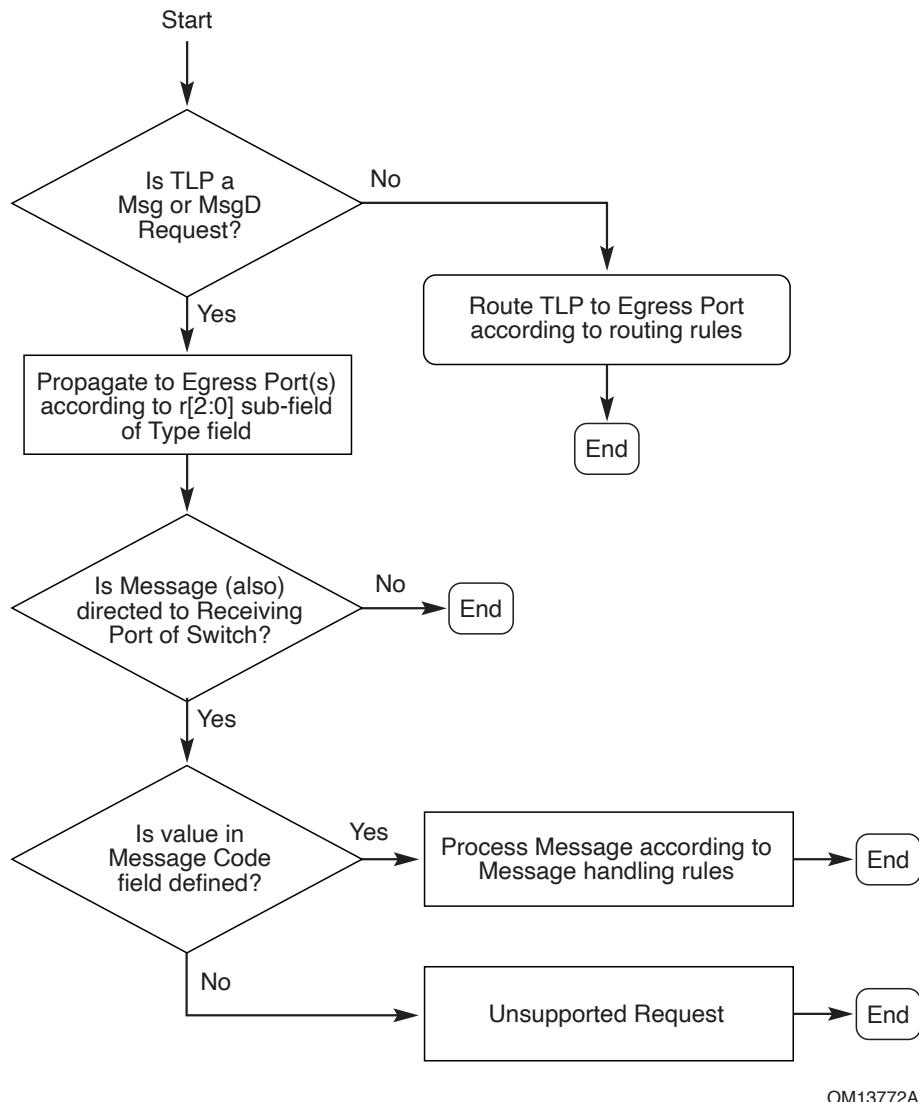
Figure 2-89 Flowchart for Handling of Received TLPs

Switches must process both TLPs that address resources within the Switch as well as TLPs that address resources residing outside the Switch. Switches handle all TLPs that address internal resources of the Switch according to the rules above. TLPs that pass through the Switch, or that address the Switch as well as passing through it, are handled according to the following rules (see § Figure 2-90):

- If the value in the Type field indicates the TLP is not a Msg or MsgD Request, the TLP must be routed according to the routing mechanism used (see § Section 2.2.4.1 and § Section 2.2.4.2).
- Switches route Completions using the information in the Requester ID field of the Completion.
- If the value in the Type field indicates the TLP is a Msg or MsgD Request, route the Request according to the routing mechanism indicated in the r[2:0] sub-field of the Type field.
 - If the value in r[2:0] indicates the Msg/MsgD is ~~not~~ Routed to the Root Complex (000b), the Switch must route the Msg/MsgD to the Upstream Port of the Switch.
 - It is an error to receive a Msg/MsgD Request specifying 000b routing at the Upstream Port of a Switch. Switches may check for violations of this rule - TLPs in violation are Malformed TLPs. If checked, this is a reported error associated with the Receiving Port (see § Section 6.2).

- If the value in r[2:0] indicates the Msg/MsgD is routed by address (001b), the Switch must route the Msg/MsgD in the same way it would route a Memory Request by address.
- If the value in r[2:0] indicates the Msg/MsgD is routed by ID (010b), the Switch must route the Msg/MsgD in the same way it would route a Completion by ID.
- If the value in r[2:0] indicates the Msg/MsgD is a broadcast from the Root Complex (011b), the Switch must route the Msg/MsgD to all Downstream Ports of the Switch.
 - It is an error to receive a Msg/MsgD Request specifying 011b routing at the Downstream Port of a Switch. Switches may check for violations of this rule - TLPs in violation are Malformed TLPs. If checked, this is a reported error associated with the Receiving Port (see § Section 6.2).
- If the value in r[2:0] indicates the Msg/MsgD terminates at the Receiver (100b or a Reserved value), or if the Message Code field value is defined and corresponds to a Message that must be comprehended by the Switch, the Switch must process the Message according to the Message processing rules.
- If the value in r[2:0] indicates Gathered and routed to Root Complex (101b), see § Section 5.3.3.2.1 for Message handling rules.
 - It is an error to receive any Msg/MsgD Request other than a PME_TO_Ack that specifies 101b routing.
 - It is an error to receive a PME_TO_Ack at the Upstream Port of a Switch. Switches may optionally check for violations of these rules. These checks are independently optional (see § Section 6.2.3.4). If checked, violations are Malformed TLPs, and are reported errors associated with the Receiving Port (see § Section 6.2).

Base 6.4 vs Base 6.3



OM13772A

Figure 2-90 Flowchart for Switch Handling of TLPs §

2.3.1 Request Handling Rules §

This section describes how Received Requests are handled, following the initial processing done with all TLPs. The rules are diagrammed in the flowchart shown in [§ Figure 2-91](#).

- If the Request Type is defined in [§ Table 2-2](#), and [§ Table 2-3](#) (NFM), or is a non-Reserved value in [§ Table 2-5](#) (FM), but is not supported (by design or because of configuration settings) by the device, the Request must be handled as an Unsupported Request (UR). This is an error associated with the Receiving Port (see [§ Section 6.2](#))
 - If the Request requires a Completion, a Completion Status of UR must be returned (see [§ Section 2.2.9](#)).
- For a Receiver that decodes UIO Type values, if the Request is a UIO Request, but the Receiver cannot forward or process the Request, then the Request must be dropped.

- In Flit Mode, if a Receiver is targeted by a Request TLP, and the Type value is defined in § Table 2-5 as a Reserved value, it is strongly recommended³⁷ for the Request to be discarded following the update of flow control credits.
- In Flit Mode, if a Routing Element receives a Request TLP to be forwarded with the Type value defined in § Table 2-5 as a Reserved value, but is unable to forward it due to a problem like the Egress Port being in DL_Down, it is strongly recommended³⁸ for the Request to be discarded following the update of flow control credits.

IMPLEMENTATION NOTE: WHEN REQUESTS ARE TERMINATED USING UNSUPPORTED REQUEST §

In Conventional PCI, a device “claims” a request on the bus by asserting DEVSEL#. If no device claims a request after a set number of clocks, the request is terminated as a Master Abort. Since PCI Express is a point to point interconnect, there is no equivalent mechanism for claiming a request on a Link, since all transmissions by one component are always sent to the other component on the Link. Therefore, it is necessary for the receiver of a request to determine if the request should be claimed. If the request is not claimed, then it is handled as an Unsupported Request, which is the PCI Express equivalent of Conventional PCI's Master Abort termination. In general, one can determine the correct behavior by asking the question: *Would the device assert DEVSEL# for this request in conventional PCI?*

For device Functions with Type 0 headers (all types of Endpoints), it is relatively simple to answer this question. For Memory and I/O Requests, this determination is based on the address ranges the Function has been programmed to respond to. For Configuration requests, the Type 0 request format indicates the device is by definition the “target”, although the device will still not claim the Configuration Request if it addresses an unimplemented Function.

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

PCI Express Messages do not exist in conventional PCI, so the above guideline cannot be applied. This specification describes specifically for each type of Message when a device must handle the request as an Unsupported Request. Messages pass through Root and Switch Ports unaffected by conventional PCI control mechanisms including **↓↑Bus Master Enable↓↑the Bus Master Enable bit↑** and power state setting.

Note that CA, which is the PCI Express equivalent to Target Abort, is used only to indicate a serious error that makes the Completer permanently unable to respond to a request that it would otherwise have normally responded to. Since Target Abort is used in conventional PCI only when a target has asserted DEVSEL#, it is incorrect to use a CA for any case where a Conventional PCI target would have ignored a request by not asserting DEVSEL#.

- If the Request is a Message, and the Message Code, routing field, or Msg / MsgD indication corresponds to a combination that is undefined, or that corresponds to a Message not supported by the device Function (other than Vendor-Defined Type 1, which is not treated as an error - see § Table F-1), the Request is an Unsupported

37. For backward compatibility with previous versions of this specification, the Request is permitted to be handled as an Unsupported Request.

38. For backward compatibility with previous versions of this specification, the Request is permitted to be handled as an Unsupported Request.

Request (UR). This is a reported error associated with the Receiving Port and is reported according to § [Section 6.2](#)

- If the Message Code is a supported value, process the Message according to the corresponding Message processing rules; if the Message Code is an Ignored Message and the Receiver is ignoring it, ignore the Message without reporting any error (see § [Section 2.2.8.7](#))
- If the Request is a Message with a routing field that indicates Routed by ID, and if the Request is received by a device Function with Type 0 headers, the device *MUST*@FLIT be treated as the target of the Message regardless of the Bus Number and, for non-ARI Devices, the Device Number specified in the Destination ID field of the Request
 - If the Function specified in the Destination ID field is unimplemented, then it is ~~↓implementation-specific~~ ↓implementation specific whether that Message is either silently discarded or else handled as an Unsupported Request (UR) and reported as an error associated with the Recieving Port (see § [Section 6.2](#)).
 - Earlier versions of this specification recommended ignoring the Bus Number and Device Number fields with the Destination ID comparison in certain cases. While this behavior is still permitted, it is no longer recommended.

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

- If the Request violates the programming model of the device Function, the Function may optionally treat the Request as a Completer Abort, instead of handling the Request normally
 - If the Request is treated as a Completer Abort, this is a reported error associated with the Function (see § [Section 6.2](#))
 - If the Request requires Completion, a Completion Status of CA is returned (see § [Section 2.2.9](#))

IMPLEMENTATION NOTE: OPTIMIZATIONS BASED ON RESTRICTED PROGRAMMING MODEL §

When a device's programming model restricts (versus what is otherwise permitted in PCI Express) the characteristics of a Request, that device is permitted to return a UR or a CA Completion Status, or to terminate the Request in a suitable ~~↓implementation-specific~~ ↓implementation specific way for any Request that violates the programming model. Examples include unaligned or wrong-size access to a register block and unsupported size of request to a Memory Space.

Generally, devices are able to rely on a restricted programming model when all communication will be between the device's driver software and the device itself. Devices directly accessed via other software (e.g., operating system, application software) may not be able to rely on a restricted programming.

Devices that implement legacy capabilities should be designed to support all types of Requests that are possible in the existing usage model for the device. If this is not done, the device may fail to operate with existing software.

If the Request arrives between the time an FLR has been initiated and the completion of the FLR by the targeted Function, the Request is permitted to be silently discarded (following update of flow control credits) without logging or signaling it as an error. It is recommended that the Request be handled as an Unsupported Request (UR).

- For DMWr Requests, refer to the rules in § Section 6.32 .
- Otherwise (supported Request Type, not a DMWr, not a Message), process the Request
 - If the Completer is permanently unable to process the Request due to a device-specific error condition the Completer must, if possible, handle the Request as a Completer Abort
 - This is a reported error associated with the Receiving Function, if the error can be isolated to a specific Function in the component, or to the Receiving Port if the error cannot be isolated (see § Section 6.2)
 - For Configuration Requests, if Device Readiness Status (DRS) is supported³⁹ , then:
 - Following any DRS Event (see § Section 6.22), once the Link is in L0, a Function associated with an Upstream Port *MUST@FLIT* return a Completion Status of RRS until the Upstream Port has transmitted a Device Readiness Status Message.
 - Once the Upstream Port has transmitted a Device Readiness Status Message, all non-VF Functions of the Upstream Port must respond to all properly formed Configuration Requests with a Completion Status of Successful Completion.
 - For Configuration Requests only, if Device Readiness Status is not supported, following reset it is permitted for a Function to terminate the request and indicate that it is temporarily unable to process the Request, but will be able to process the Request in the future - in this case, the Request Retry Status (RRS) Completion Status must be used (see § Section 6.6). Valid reset conditions after which a device/Function is permitted to return RRS in response to a Configuration Request are:
 - Cold, Warm, and Hot Resets
 - FLRs
 - A reset initiated in response to a D3Hot to D0uninitialized device state transition
 - A device Function is explicitly not permitted to return RRS in response to a Configuration Request following a software-initiated reset (other than an FLR) of the device, e.g., by the device's software driver writing to a device-specific reset bit. A device Function is not permitted to return RRS in response to a Configuration Request after it has indicated that it is Configuration-Ready (see § Section 6.22) without an intervening valid reset (i.e., FLR or Conventional Reset) condition, or if the Immediate Readiness bit in the Function's Status register is Set. Additionally, a device Function is not permitted to return RRS in response to a Configuration Request after having previously returned a Successful Completion without an intervening valid reset (i.e., FLR or Conventional Reset) condition.
 - A Function that implements the Readiness Time Reporting Extended Capability must not return RRS in response to Configuration Requests that are received after the relevant times reported in that Extended Capability.
 - 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↓↓↑↑error.↑**
 - Example: A PCI Express/PCI Bridge may initially accept a Request because it specifies a Memory Space range mapped to the secondary side of the Bridge, but the Request may Master Abort or Target Abort on the PCI side of the Bridge. From the PCI Express perspective, the status of the Request in this case is UR (for Master Abort) or CA (for Target Abort). If the Request requires Completion on PCI Express, the corresponding Completion Status is returned.
- If the Request is a type that requires a Completion to be returned, generate a Completion according to the rules for Completion formation (see § Section 2.2.9)
 - The Completion Status is determined by the result of handling the Request

39. If Flit Mode is supported then Device Readiness Status must be supported.

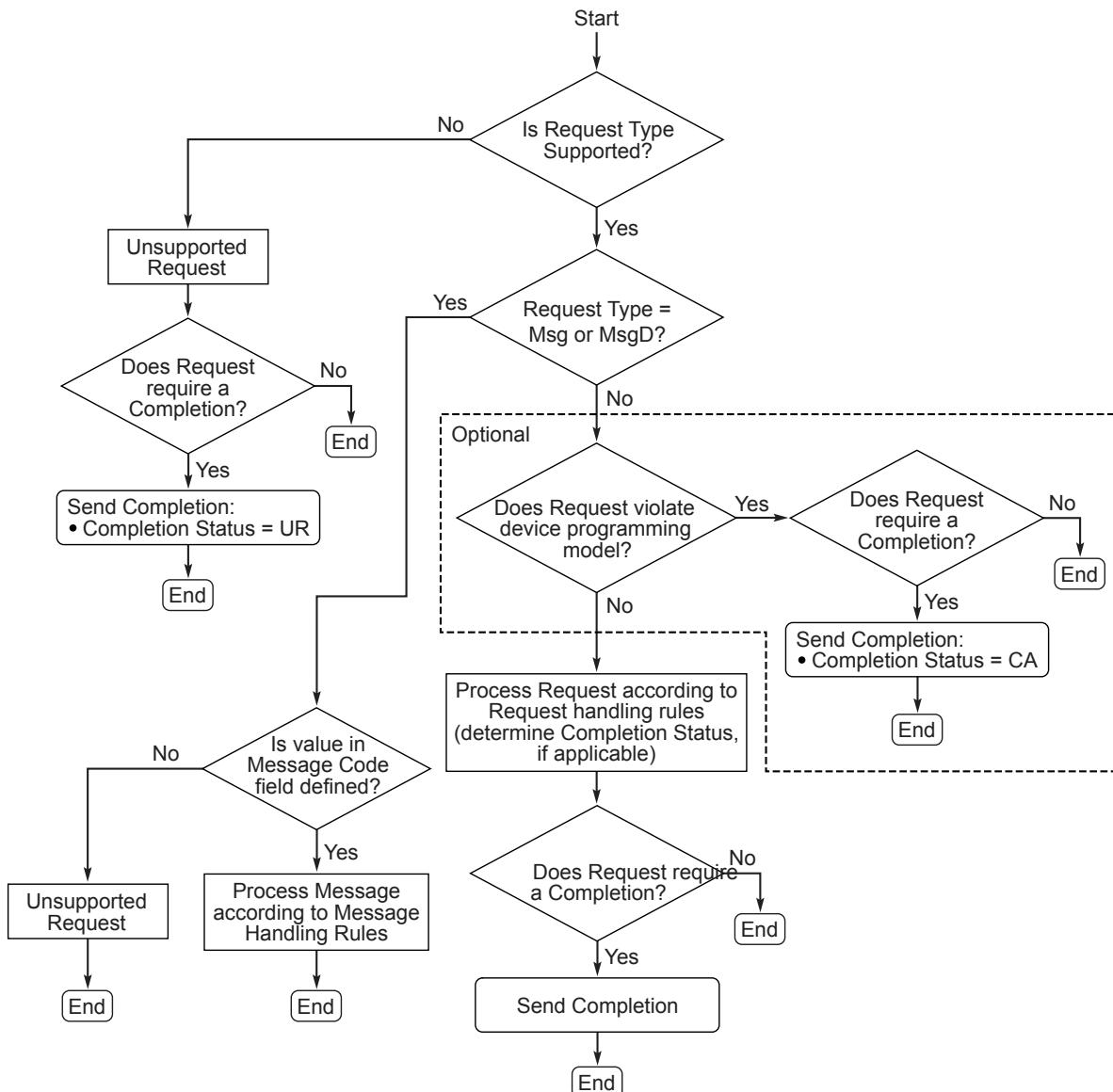
- If the Request has an ECRC Check Failed error, then it is implementation specific whether to return a Completion or not. If a Completion is returned, the Completion *MUST* have a UR Completion Status.
- Under normal operating conditions, PCI Express Endpoints and Legacy Endpoints must never delay the acceptance of a Posted Request for more than 10 µs, which is called the ***Posted Request Acceptance Limit***. The device must either (a) be designed to process received Posted Requests and return associated Flow Control credits within the necessary time limit, or (b) rely on a restricted programming model to ensure that a Posted Request is never initiated to the Endpoint either by software or by other devices while the Endpoint is unable to accept a new Posted Request within the necessary time limit.
 - The following are not considered normal operating conditions under which the Posted Request Acceptance Limit applies:
 - The period immediately following a Fundamental Reset (see § [Section 6.6](#))
 - TLP retransmissions or Link retraining
 - One or more dropped Flow Control Packets (FCPs)
 - The device being in a diagnostic mode
 - The device being in a device-specific mode that is not intended for normal use
 - The following are considered normal operating conditions, but any delays they cause do not count against the Posted Request Acceptance Limit :
 - Upstream TLP traffic delaying Upstream FCPs
 - The Link coming out of a low-power state
 - Arbitration with traffic on other VCs
 - Though not a requirement, it is strongly recommended that RCiEPs also honor the Posted Request Acceptance Limit.
- If the device/Function supports being a target for I/O Write Requests, which are Non-Posted Requests, it is strongly recommended that each associated Completion be returned within the same time limit as for Posted Request acceptance, although this is not a requirement.
- If the device/Function supports being a target for DMWr Requests, each associated Completion must be returned within the Posted Request Acceptance Limit⁴⁰

IMPLEMENTATION NOTE: RESTRICTED PROGRAMMING MODEL FOR MEETING THE POSTED REQUEST ACCEPTANCE LIMIT §

Some hardware designs may not be able to process every DMWr or Posted Request within the required Posted Request Acceptance Limit. An example is writing to a command queue where commands can take longer than the acceptance time limit to complete. Subsequent writes to such a device when it is currently processing a previous write could experience acceptance delays that exceed the limit. Such devices may rely on a restricted programming model, where the device driver limits the rate of DMWr/memory writes issued to the device, the driver polls the device to determine buffer availability before issuing the transaction, or the driver implements some other software-based flow control mechanism.

40. Although DMWr is a Non-Posted Request, the Posted Request Acceptance Limit is applied because many (but not all) of the same concerns apply as with Posted Requests. The name Posted Request Acceptance Limit is retained for historical reasons.

Base 6.4 vs Base 6.3



OM13773A

Figure 2-91 Flowchart for Handling of Received Requests

IMPLEMENTATION NOTE: REQUEST RETRY STATUS FOR CONFIGURATION REQUESTS §

Some devices require a lengthy self-initialization sequence to complete before they are able to service Configuration Requests. In specified circumstances it is permitted for a device/Function to “hold off” initial configuration via the Request Retry Status (RRS) Completion Status mechanism. A device in receipt of a Configuration Request following a valid reset condition may respond with an RRS Completion Status to terminate the Request, and thus effectively stall the Configuration Request until such time that the subsystem has completed local initialization and is ready to communicate with the host. Note that it is only legal in specified circumstances to respond with an RRS Completion Status in response to a Configuration Request. Readiness Notifications (see § Section 6.22) and Immediate Readiness (see § Section 7.5.1.1.4 ↑↑↑ and ↑↑↑Immediate Readiness on Return to D0 (see § Section 7.5.2.1) also forbid the use of RRS Completion Status in response to a Configuration Request in certain situations.

Receipt by the Requester of a Completion with RRS Completion Status terminates the Configuration Request. Further action by the Root Complex regarding the original Configuration Request is specified in § Section 2.3.2 .

Root Complexes that implement Configuration RRS Software Visibility have the ability to report the receipt of RRS Completion Status for a Configuration Request to software, enabling software to attend to other tasks rather than being stalled while the device completes its self-initialization. Software that intends to take advantage of this mechanism must ensure that the first access made to a device following a valid reset condition is a Configuration Read Request accessing both bytes of the Vendor ID field in the device's Configuration Space header. For this case only, the Root Complex, if enabled, will synthesize a special read-data value for the Vendor ID field to indicate to software that RRS Completion Status has been returned by the device in response to a Configuration Request. For Configuration Requests to other addresses, or when Configuration RRS Software Visibility is not enabled, the Root Complex will generally re-issue the Configuration Request until it completes with a status other than RRS as described in § Section 2.3.2 .

Systems that contain PCIe components whose self-initialization time may require them to return a RRS Completion Status in response to a Configuration Request (by the rules in § Section 6.6) should provide some mechanism for re-issuing Configuration Requests terminated with RRS status. In systems running legacy PCI/PCI-X based software, the Root Complex must re-issue the Configuration Request using a hardware mechanism to ensure proper enumeration of the system.

Refer to § Section 6.6 for more information on reset.

2.3.1.1 Data Return for Non-UIO 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.
 - A Completion with Completion Status other than Successful Completion must:
 - be of type Cpl or CplLk,
 - for a Read Request, including ↓↓an ATS↓ ↑↑↑ Translation Request, be the final Completion returned.
- Completions must not include more data than permitted by the Transmitting Function's Tx_MPS_Limit .

- A Receiving Function must check for violations of its Rx_MPS_Limit.
- See § Section 2.2.2 for important details with both Transmitters and Receivers.

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

- Memory Read Requests may be completed with one, or in some cases, multiple Completions
- Read Completion Boundary (RCB) determines the naturally aligned address boundaries on which a Completer is permitted to break up the response for a single Read Request into multiple Completions.
 - For a Root Complex, RCB is 64 bytes or 128 bytes.
 - This value is reported in the Read Completion Boundary     field in the Link Control Register (see § Section 7.5.3.7).

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

- For all other System Elements, RCB is 128 bytes.
- Completions for Requests that do not cross the naturally aligned address boundaries at integer multiples of RCB bytes must include all data specified in the Request.
- Requests that do cross the address boundaries at integer multiples of RCB bytes are permitted to be completed using more than one Completion subject to the following rules:
 - The first Completion must start with the address specified in the Request, and if successful must end at one of the following:
 - The address that satisfies the entire Request
 - An address boundary between the start and end of the Request at an integer multiple of RCB bytes
 - If the final Completion is successful, it must end at the address that satisfies the entire Request
 - All Completions between, but not including, the first and final Completions must be an integer multiple of RCB bytes in length
- Receivers may optionally check for violations of RCB. If a Receiver implementing this check determines that a Completion violates this rule, it must handle the Completion as a Malformed TLP.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).
- Multiple Memory Read Completions for a single Read Request must return data in increasing address order.
- If all the Memory Read Completions for a single Read Request have a Successful Completion Status, the sum of their payloads must equal the size requested. See § Section 10.2.4 for an exception for Memory Reads that are ATS Translation Requests.
- For each Memory Read Completion, the Byte Count field must indicate the remaining number of bytes required to complete the Request including the number of bytes returned with the Completion, except when the BCM bit is Set.⁴¹
 - The total number of bytes required to complete a Memory Read Request is calculated as shown in   Table 2-40.
 - If a Memory Read Request is completed using multiple Completions, the Byte Count value for each successive Completion is the value indicated by the preceding Completion minus the number of bytes returned with the preceding Completion.

41. Only PCI-X completers Set the BCM bit. PCI Express completers are not permitted to set the BCM bit. In Flit Mode, the BCM bit is deprecated. When translating from Non-Flit Mode to Flit Mode, if the BCM bit is 1b a TLP Translation Egress Blocked error must be indicated. When translating from Flit Mode to Non-Flit Mode the BCM bit must be Clear in the Non-Flit Mode Header.

- The Completion Data area begins at the DW address specified by the Request. In the first or only Data DW of the first or only Completion, only the bytes configured as active in the First DW BE field in the Request contain valid data. Bytes configured as inactive in the First DW BE field in the Request will return undefined content.
- In the last Data DW of the last successful Completion, only the bytes configured as active in the Last DW BE field in the Request contain valid data. Bytes configured as inactive in the Last DW BE field in the Request will return undefined content.
- All the Completion Data bytes, including those with undefined content, are included in all CRC calculations.
- § Figure 2-92 presents an example of the above. The example assumes a single Completion TLP is returned.

Base 6.4 vs Base 6.3

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

Length = 4
Byte Count = 10

Figure 2-92 Example Completion Data when some Byte Enables are 0b §

IMPLEMENTATION NOTE: BCM BIT USAGE §

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

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

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

PCI Express Completers will never Set the BCM bit.

The BCM bit is not present in Flit Mode.

Base 6.4 vs Base 6.3

*Table 2-40 Calculating Byte Count from Length and
First/Last Byte Enables §*

| First DW BE[3:0] (b) | Last DW BE[3:0] (b) | Total Byte Count |
|----------------------|---------------------|--------------------------|
| 1xx1 | 0000 ⁴² | 4 |
| 01x1 | 0000 | 3 |
| 1x10 | 0000 | 3 |
| 0011 | 0000 | 2 |
| 0110 | 0000 | 2 |
| 1100 | 0000 | 2 |
| 0001 | 0000 | 1 |
| 0010 | 0000 | 1 |
| 0100 | 0000 | 1 |
| 1000 | 0000 | 1 |
| 0000 | 0000 | 1 |
| xxx1 | 1xxx | Length ⁴³ * 4 |
| xxx1 | 01xx | (Length * 4) - 1 |
| xxx1 | 001x | (Length * 4) - 2 |
| xxx1 | 0001 | (Length * 4) - 3 |
| xx10 | 1xxx | (Length * 4) - 1 |
| xx10 | 01xx | (Length * 4) - 2 |
| xx10 | 001x | (Length * 4) - 3 |
| xx10 | 0001 | (Length * 4) - 4 |
| x100 | 1xxx | (Length * 4) - 2 |
| x100 | 01xx | (Length * 4) - 3 |
| x100 | 001x | (Length * 4) - 4 |
| x100 | 0001 | (Length * 4) - 5 |
| 1000 | 1xxx | (Length * 4) - 3 |
| 1000 | 01xx | (Length * 4) - 4 |
| 1000 | 001x | (Length * 4) - 5 |
| 1000 | 0001 | (Length * 4) - 6 |

42. Note that Last DW BE of 0000b is permitted only with a Length of 1 DW.

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

- For all Memory Read Completions, the Lower Address field must indicate the lower bits of the byte address for the first enabled byte of data returned with the Completion.
 - For the first (or only) Completion, the Completer can generate this field from the least significant 5 bits of the address of the Request concatenated with 2 bits of byte-level address formed as shown in [Table 2-41](#).
 - For any subsequent Completions, the Lower Address field will always be zero except for Completions generated by a Root Complex with an RCB value of 64 bytes. In this case the least significant 6 bits of the Lower Address field will always be zero and the most significant bit of the Lower Address field will toggle according to the alignment of the 64-byte data payload.

*Table 2-41 Calculating Lower Address from
First DW $\downarrow\downarrow BE$ $\uparrow\uparrow$ Byte Enables*

| First DW BE[3:0] (b) | Lower Address[1:0] (b) |
|----------------------|------------------------|
| 0000 | 00 |
| xxx1 | 00 |
| xx10 | 01 |
| x100 | 10 |
| 1000 | 11 |

- When a Read Completion is generated with a Completion Status other than Successful Completion:
 - No data is included with the Completion
 - The Cpl (or CplLk) encoding is used instead of CplD (or CplDLk)
 - This Completion is the final Completion for the Request
 - The Completer must not transmit additional Completions for this Request
 - Example: Completer split the Request into four parts for servicing; the second Completion had a Completer Abort Completion Status; the Completer terminated servicing for the Request, and did not Transmit the remaining two Completions.
 - The Byte Count field must indicate the remaining number of bytes that would be required to complete the Request (as if the Completion Status were Successful Completion)
 - The Lower Address field must indicate the lower bits of the byte address for the first enabled byte of data that would have been returned with the Completion if the Completion Status were Successful Completion

IMPLEMENTATION NOTE: RESTRICTED PROGRAMMING MODEL §

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

Examples:

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

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

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

256 -or-

32, 224 -or- 32, 64, 160 -or- 32, 64, 64, 96 -or- 32, 64, 64, 64, 32 -or-

32, 64, 128, 32 -or- 32, 128, 96 -or- 32, 128, 64, 32 -or-

96, 160 -or- 96, 128, 32 -or- 96, 64, 96 -or- 96, 64, 64, 32 -or-

160, 96 -or- 160, 64, 32 -or- 224, 32
4. Memory Read Request with Address of 1 0020h and Length of 100h bytes (256 decimal) could be completed by an Endpoint in one of the following combinations of Completions (bytes):

256 -or- 96, 160 -or- 96, 128, 32 -or- 224, 32

2.3.1.2 UIO Read Completions §

- UIO Read Completions must follow the same rules as non-UIO Read Completions, with the following exceptions:
 - Multiple UIO Read Completions for a single UIO Read Request are permitted to be returned in any address order.

2.3.1.3 UIO Write Completions §

- UIO Write Completions must follow these rules:
 - Write Completion Boundary (WCB) is 64B, and indicates the naturally aligned address boundaries on which a Completer is permitted to break up the response for a single UIO Write Request into multiple Completions.
 - Multiple UIO Write Completions for a single UIO Write Request are permitted to be returned in any address order.
- It is permitted for UIO Completers to coalesce UIO Write Completions with the same Transaction ID.
 - Completion coalescing must not exceed the number of DWORDs that can be represented in the Length[9:0] field.
 - It is recommended that UIO Completers coalesce opportunistically, and, in most cases, it is recommended not to delay the return of a UIO Completion in order to coalesce it with a subsequent UIO Completion. Specific policies for UIO Completion coalescing are implementation-specific.
- Switches are not permitted to coalesce UIO Completions.

2.3.2 Completion Handling Rules §

- When a device receives a Completion that does not match the Transaction ID for any of the outstanding Requests issued by that device, the Completion is called an “Unexpected Completion”.
- If a received Completion matches the Transaction ID of an outstanding Request, but in other TLP fields does not match the corresponding Request, the Receiver *MUST@FLIT* handle the Completion as an Unexpected Completion.
 - The Requester must not check the IDO Attribute (Attribute Bit 2) in the Completion, since the Completer is not required to copy the value of IDO from the Request into the Completion for that request as stated in § [Section 2.2.6.4](#) and § [Section 2.2.9](#).
 - However, if the Completion is otherwise properly formed, it is permitted for the Receiver to handle the Completion as ~~↑↓anv~~ ↑↓a↑ Malformed TLP.
- When an Ingress Port of a Switch receives a Completion that cannot be forwarded, that Ingress Port must handle the Completion as an Unexpected Completion. This includes Completions that target:
 - a non-existent Function in the Device associated with the Upstream Port,
 - a non-existent Device on the Bus associated with the Upstream Port,
 - a non-existent Device or Function on the internal switching fabric, or
 - a Bus Number within the Upstream Port's Bus Number aperture but not claimed by any Downstream Port.
- Receipt of an Unexpected Completion is an error and must be handled according to the following rules:
 - The agent receiving an Unexpected Completion must discard the Completion.
 - An Unexpected Completion is a reported error associated with the Receiving Port (see § [Section 6.2](#)).

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

- Completions with a Completion Status other than Successful Completion or Request Retry Status (in response to Configuration Request) must cause the Requester to:
 - Free Completion buffer space and other resources associated with the Request.
 - Handle the error via a Requester-specific mechanism (see § [Section 6.2.3.2.5](#)).

If the Completion arrives between the time an FLR has been initiated and the completion of the FLR by the targeted Function, the Completion is permitted to be handled as an Unexpected Completion or to be silently discarded (following update of flow control credits) without logging or signaling it as an error. Once the FLR has completed, received Completions corresponding to Requests issued prior to the FLR must be handled as Unexpected Completions, unless the Function has been re-enabled to issue Requests.

- Root Complex handling of a Completion with Request Retry Status for a Configuration Request is implementation specific, except for the period following system reset (see § [Section 6.6](#)). For Root Complexes that support Configuration RRS Software Visibility, the following rules apply:
 - If Configuration RRS Software Visibility is not enabled, the Root Complex must re-issue the Configuration Request as a new Request.
 - If Configuration RRS Software Visibility is enabled (see below):
 - For a Configuration Read Request that includes both bytes of the Vendor ID field of a device Function's Configuration Space Header, the Root Complex must complete the Request to the host by returning a read-data value of 0001h for the Vendor ID field and all 1's for any additional bytes included in the request. This read-data value has been reserved specifically for this use by the PCI-SIG and does not correspond to any assigned Vendor ID.
 - For a Configuration Write Request or for any other Configuration Read Request, the Root Complex must re-issue the Configuration Request as a new Request.

A Root Complex implementation may choose to limit the number of Configuration Request/RRS Completion Status loops before determining that something is wrong with the target of the Request and taking appropriate action (e.g., complete the Request to system software as a failed transaction).

Configuration RRS Software Visibility may be enabled through the Configuration RRS Software Visibility Enable bit in the Root Control Register (see § [Section 7.5.3.12](#)) to control Root Complex behavior on an individual Root Port basis. Alternatively, Root Complex behavior may be managed through the Configuration RRS Software Visibility Enable bit in the Root Complex Register Block (RCRB) Control register as described in § [Section 7.9.7.4](#) , permitting the behavior of one or more Root Ports or RCiEPs to be controlled by a single Enable bit. For this alternate case, each Root Port or RCiEP declares its association with a particular Enable bit via an RCRB header association in a Root Complex Link Declaration Capability (see § [Section 7.9.8](#)). Each Root Port or RCiEP is permitted to be controlled by at most one Enable bit. Thus, for example, it is prohibited for a Root Port whose Root Control register contains an Enable bit to declare an RCRB header association to an RCRB that also includes an Enable bit in its RCRB Header Capability. The presence of an Enable bit in a Root Port or RCRB Header Capability is indicated by the corresponding Configuration RRS Software Visibility bit (see § [Section 7.5.3.13](#) and § [Section 7.9.7.3](#) , respectively).

- Completions with a Reserved Completion Status value are treated as if the Completion Status was Unsupported Request (UR).
- Completions with a Completion Status of Unsupported Request or Completer Abort are reported using the conventional PCI reporting mechanisms (see § [Section 7.5.1.1.4](#)).
 - Note that the error condition that triggered the generation of such a Completion is reported by the Completer as described in § [Section 6.2](#) .
- When a Completion for a Read, AtomicOp, DMWr Request 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 CplID (CplDLk)
- This Completion is the final Completion for the Request
 - The Requester must consider the Request terminated, and not expect additional Completions
 - Handling of partial Completions Received earlier is implementation specific

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

IMPLEMENTATION NOTE: READ DATA VALUES WITH UR COMPLETION STATUS §

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

2.4 Transaction Ordering §

2.4.1 Transaction Ordering Rules for TLPs not using UIO or Flow-Through IDE Streams §

The rules defined in this section apply uniformly to all types of Transactions on PCI Express including Memory, I/O, Configuration, and Messages, except for UIO Requests/Completions (see § Section 2.4.2) and Flow-Through IDE Streams which have modified ordering requirements (see [§ Section 6.33.7](#)). The ordering rules defined in this table apply within a single Traffic Class (TC). There is no ordering requirement among transactions with different TC labels. Note that this also implies that there is no ordering required between traffic that flows through different Virtual Channels since transactions with the same TC label are not allowed to be mapped to multiple VCs on any PCI Express Link.

For [§ Table 2-42](#) , the columns represent a first issued transaction and the rows represent a subsequently issued transaction. The table entry indicates the ordering relationship between the two transactions. The table entries are defined as follows:

Yes

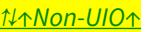
The second transaction (row) must be allowed to pass the first (column) to avoid deadlock. (When blocking occurs, the second transaction is required to pass the first transaction. Fairness must be comprehended to prevent starvation.)

Y/N

There are no requirements. The second transaction may optionally pass the first transaction or be blocked by it.

No

The second transaction must not be allowed to pass the first transaction. This is required to support the producer/consumer strong ordering model.

Table 2-42  

| Row Pass Column? | | Posted Request (Col 2) | Non-Posted Request | | Completion (Col 5) |
|--------------------|---------------------------|---------------------------|-------------------------|--------------------------|-------------------------|
| | | | Read Request (Col 3) | NPR with Data (Col 4) | |
| | Posted Request (Row A) | a) No <u>b) Y/N</u> | <u>Yes</u> | <u>Yes</u> | a) Y/N <u>b) Yes</u> |
| Non-Posted Request | Read Request (Row B) | a) No <u>b) Y/N</u> | <u>Y/N</u> | <u>Y/N</u> | <u>Y/N</u> |
| | NPR with Data (Row C) | a) No <u>b) Y/N</u> | <u>Y/N</u> | <u>Y/N</u> | <u>Y/N</u> |
| | Completion (Row D) | a) No <u>b) Y/N</u> | <u>Yes</u> | <u>Yes</u> | a) Y/N <u>b) No</u> |

Explanation of the row and column headers in  :

A **Posted Request** is a Memory Write Request or a Message Request.

A **Read Request** is a Configuration Read Request, an I/O Read Request, or a Memory Read Request.

An **NPR** (Non-Posted Request) **with Data** is a Configuration Write Request, an I/O Write Request, an AtomicOp Request, or a DMWr.

A **Non-Posted Request** is a Read Request or an NPR with Data.

Explanation of the entries in  :

A2a

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

A2b

A Posted Request with RO⁴⁴ Set is permitted to pass another Posted Request.⁴⁵ A Posted Request with IDO Set is permitted to pass another Posted Request if the two Requester IDs (including Requester Segment when in FM) are different. Additionally, a Posted Request with IDO Set is permitted to pass another Posted Request with the same Requester ID if both Requests contain a PASID and the two PASID values are different.

A3 , A4

A Posted Request must be able to pass Non-Posted Requests to avoid deadlocks.

A5a

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

A5b

Inside a PCI Express to PCI/PCI-X Bridge whose PCI/PCI-X bus segment is operating in conventional PCI mode, for transactions traveling in the PCI Express to PCI direction, a Posted Request must be able to pass Completions to avoid deadlock.

B2a

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

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

45. Some usages are enabled by not implementing this passing (see the No RO-enabled PR-PR Passing bit in § Section 7.5.3.15).

B2b

A Read Request with IDO Set is permitted to pass a Posted Request if the two Requester IDs (including Requester Segment in FM) are different. Additionally, a Read Request with IDO Set is permitted to pass a Posted Request with the same Requester ID if both Requests contain a PASID and the two PASID values are different.

C2a

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

C2b

An NPR with Data and with RO Set⁴⁶ is permitted to pass Posted Requests. An NPR with Data and with IDO Set is permitted to pass a Posted Request if the two Requester IDs (including Requester Segment in FM) are different. Additionally, an NPR with Data and with IDO Set is permitted to pass a Posted Request with the same Requester ID if both Requests contain a PASID and the two PASID values are different.

B3 , B4 , C3 , C4

A Non-Posted Request is permitted to pass another Non-Posted Request.

B5 , C5

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

D2a

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

D2b

An I/O or Configuration Write Completion⁴⁷ is permitted to pass a Posted Request. A Completion with RO Set is permitted to pass a Posted Request. A Completion with IDO Set is permitted to pass a Posted Request if the Completer ID of the Completion is different from the Requester ID of the Posted Request.

D3 , D4

A Completion must be able to pass Non-Posted Requests to avoid deadlocks.

D5a

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

D5b

Completions with the same Transaction ID must not pass each other. This ensures that multiple Completions associated with a single Memory Read Request will remain in ascending address order.

Additional Rules:

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

46. Note: Not all NPR with Data transactions are permitted to have RO Set.

47. Note: Not all components can distinguish I/O and Configuration Write Completions from other Completions. In particular, routing elements not serving as the associated Requester or Completer generally cannot make this distinction. A component must not apply this rule for I/O and Configuration Write Completions unless it is certain of the associated Request type.

- Combining of Memory Read Requests, and/or Completions for different Requests is prohibited.
- The No Snoop bit does not affect the required ordering behavior.
- For Root Ports and Switch Downstream Ports, acceptance of a Posted Request or Completion must not depend upon the transmission of a Non-Posted Request within the same traffic class.⁴⁸
- For Switch Upstream Ports, acceptance of a Posted Request or Completion must not depend upon the transmission on a Downstream Port of Non-Posted Request within the same traffic class.⁴⁹
- For Endpoint, Bridge, and Switch Upstream Ports, the acceptance of a Posted Request must not depend upon the transmission of any TLP from that same Upstream Port within the same traffic class.⁵⁰
- For Endpoint, Bridge, and Switch Upstream Ports, the acceptance of a Non-Posted Request must not depend upon the transmission of a Non-Posted Request from that same Upstream Port within the same traffic class.⁵¹
- For Endpoint, Bridge, and Switch Upstream Ports, the acceptance of a Completion must not depend upon the transmission of any TLP from that same Upstream Port within the same traffic class.⁵²

Note that Endpoints are never permitted to block acceptance of a Completion.

- Completions issued for Non-Posted Requests must be returned in the same Traffic Class as the corresponding Non-Posted Request.
- Root Complexes that support peer-to-peer operation and Switches must enforce these transaction ordering rules for all forwarded traffic.

To ensure deadlock-free operation, devices should not forward traffic from one Virtual Channel to another. The specification of constraints used to avoid deadlock in systems where devices forward or translate transactions between Virtual Channels is outside the scope of this document (see § Appendix D. for a discussion of relevant issues).

-
48. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see § Appendix D. for a discussion of relevant issues).
 49. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see § Appendix D. for a discussion of relevant issues).
 50. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see § Appendix D. for a discussion of relevant issues).
 51. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see § Appendix D. for a discussion of relevant issues).
 52. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see § Appendix D. for a discussion of relevant issues).

IMPLEMENTATION NOTE: DEADLOCKS CAUSED BY PORT ACCEPTANCE DEPENDENCIES §

With certain configurations and communication paradigms, systems whose Ports have acceptance dependencies may experience deadlocks. In this context, Port acceptance dependencies refer to the Ingress Port making the acceptance of a Posted Request or Completion dependent upon the Egress Port first being able to transmit one or more TLPs. As stated earlier in this section, Endpoints, Bridges, and Switch Upstream Ports are forbidden to have these dependences. However, Downstream Ports are allowed to have these dependencies.

In certain cases, Downstream Port acceptance dependencies are unavoidable. For example, the ACS P2P Request Redirect mechanism may be redirecting some peer-to-peer Posted Requests Upstream through an RP for validation in the Root Complex. Validated Posted Requests are then reflected back down through the same RP so they can make their way to their original targets. The validated Posted Requests set up the acceptance dependency due to this internal looping. For traffic within one system, Downstream Port acceptance dependencies do not contribute to deadlocks. However, for certain types of traffic between systems, Downstream Port acceptance dependencies can contribute to deadlocks.

One general case where this may contribute to deadlocks is when two or more systems have an interconnect that enables each host to map host memory in other hosts for Programmed I/O (PIO) access, as shown on the left side of [Figure x](#). A specific example is when two systems each have a PCIe Switch with one or more integrated by Non-Transparent Bridges (NTBs), and the two systems are connected as shown on the right side of the figure.

Deadlock can occur if each host CPU is doing a heavy stream of Posted Requests to host memory in the opposite host. If Posted Request traffic in each direction gets congested, and the [Root Port \(RP\)](#) in each host stops accepting Posted Requests because the RP can't transmit outbound TLPs, deadlock occurs. The root cause of deadlock in this case is actually the adapter to the system-to-system interconnect setting up an acceptance dependency, which is forbidden for Endpoints. For the example case of PCIe Switches with integrated NTBs, the NTBs are Endpoints, and the Switch Upstream Port has the acceptance dependency. While the [Root Port's](#) acceptance dependency is not the root cause of the deadlock, it contributes to the deadlock.

Solutions using this paradigm for intersystem communications will either need to determine that their systems don't have these acceptance dependencies or rely on other mechanisms to avoid these potential deadlocks. Such mechanisms are outside the scope of this specification.

Base 6.4 vs Base 6.3

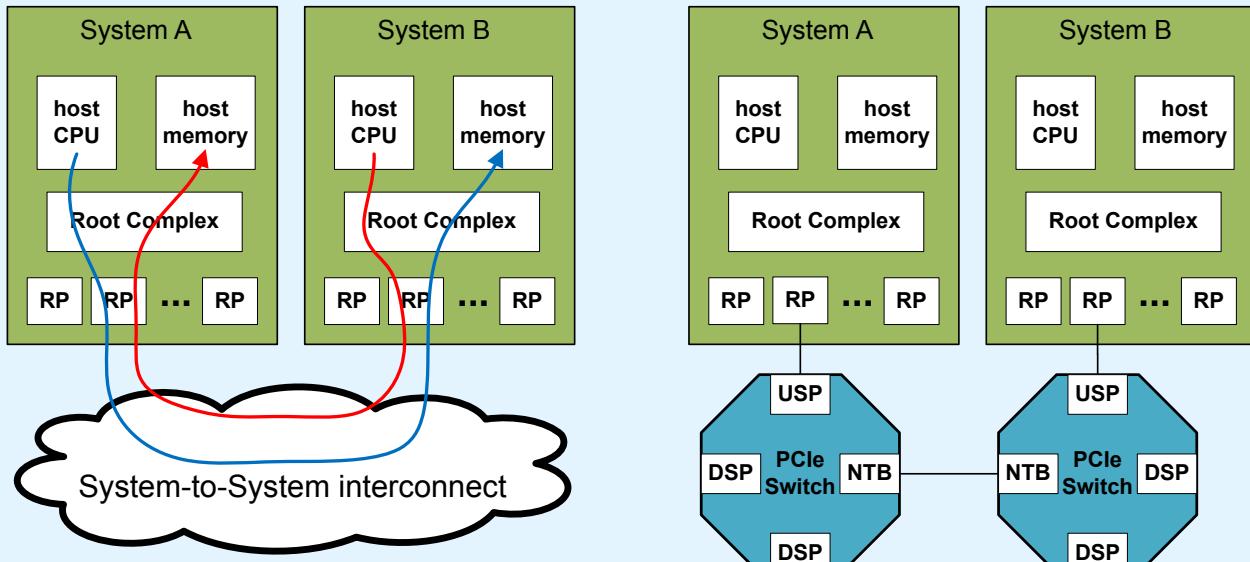


Figure 2-93 Deadlock Examples with Intersystem Interconnects §

IMPLEMENTATION NOTE: LARGE MEMORY READS VS. MULTIPLE SMALLER MEMORY READS §

Note that the rule associated with entry D5b in [Table 2-42](#) ensures that for a single Memory Read Request serviced with multiple Completions, the Completions will be returned in address order. However, the rule associated with entry D5a permits that different Completions associated with distinct Memory Read Requests may be returned in a different order than the issue order for the Requests. For example, if a device issues a single Memory Read Request for 256 bytes from location 1000h, and the Request is returned using two Completions (see [Section 2.3.1.1](#)) of 128 bytes each, it is guaranteed that the two Completions will return in the following order:

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

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

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

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

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

- or -

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

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

2.4.2 Ordering Rules for UIO §

The rules defined in this section apply to UIO TLPs. UIO and non-UIO TLPs are never mixed within a TC/VC, and ordering dependencies between UIO and non-UIO TLPs are not permitted.

The ordering rules for UIO TLPs are (see § Table 2-43):

- UIO Completions must be allowed to pass UIO Requests
- UIO allows arbitrary reordering for all other cases

Table 2-43 UIO TLP Ordering Rules §

| Row Pass Col? | UIO PR-FC TLP (Col U1) | UIO NPR-FC TLP (Col U2) | UIO Completion (Col U3) |
|----------------------------|---------------------------|----------------------------|----------------------------|
| UIO PR-FC TLP (Row UA) | Yes/No | Yes/No | Yes/No |
| UIO NPR-FC TLP (Row UB) | Yes/No | Yes/No | Yes/No |
| UIO Completion (Row UC) | Yes | Yes | Yes/No |

It is recommended that permitted UIO reordering cases be supported and implemented.

The deadlock avoidance rules for UIO TLPs are covered in § Table 2-44 and § Table 2-45.

- For considerations on acceptance dependencies, see IMPLEMENTATION NOTE: DEADLOCKS CAUSED BY PORT ACCEPTANCE DEPENDENCIES
- Downstream Ports include Root Ports and Switch Downstream Ports
- Upstream Ports include Switch Upstream Ports and Endpoint Upstream Ports

Table 2-44 UIO Acceptance Dependency Rules – Downstream Ports §

| Row Independent of Col? | | Egress transmission | | |
|-------------------------|-------------------------------------|-------------------------------------|-------------------------------------|---------------------------|
| | | UIO Memory Write Request (Col 1) | Other UIO Memory Request (Col 2) | UIO Completion (Col 3) |
| Ingress Acceptance | UIO Memory Write Request (Row A) | Yes/No | Yes/No | Yes/No |
| | Other UIO Memory Request (Row B) | Yes/No | Yes/No | Yes/No |
| | UIO Completion (Row C) | Yes | Yes | Yes/No |

Table 2-45 UIO Acceptance Dependency Rules – Upstream Ports §

| Row Independent of Col? | | Egress transmission | | |
|-------------------------|----------------------------------|----------------------------------|----------------------------------|------------------------|
| | | UIO Memory Write Request (Col 1) | Other UIO Memory Request (Col 2) | UIO Completion (Col 3) |
| Ingress Acceptance | UIO Memory Write Request (Row A) | Yes | Yes | Yes/No |
| | Other UIO Memory Request (Row B) | Yes | Yes | Yes/No |
| | UIO Completion (Row C) | Yes | Yes | Yes |

2.4.3 Update Ordering and Granularity Observed by a Read Transaction §

2.4.3.1 Ordering and Granularity for Non-UIO Reads §

If a Requester using a single transaction reads a block of data from a Completer, and the Completer's data buffer is concurrently being updated, the ordering of multiple updates and granularity of each update reflected in the data returned by the read is outside the scope of this specification, unless otherwise specified (see § Section 6.32). This applies both to updates performed by PCI Express write transactions and updates performed by other mechanisms such as host CPUs updating host memory.

If a Requester using a single transaction reads a block of data from a Completer, and the Completer's data buffer is concurrently being updated by one or more entities not on the PCI Express fabric, the ordering of multiple updates and granularity of each update reflected in the data returned by the read is outside the scope of this specification, unless otherwise specified (see § Section 6.32).

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

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

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

IMPLEMENTATION NOTE: NO ORDERING REQUIRED BETWEEN CACHELINES §

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

2.4.3.2 Ordering and Granularity for UIO Reads §

If a Requester uses a single UIO Read Request to read a block of data from a Completer, and the Completer's data buffer is concurrently being updated using UIO Write Requests, the granularity of each update reflected in the data returned by the read must, within each 64B aligned block, be observed such that each UIO Write Request is either fully completed or not-at-all completed.

Once a Completer transmits a UIO Read Completion reflecting the updated value resulting from a UIO Write Request to a 64B aligned block, subsequently received UIO Requests must observe the UIO Write as fully completed for that 64B aligned block. Thus, at any other Link or Port between a Requester and Completer, observation of a UIO Read Completion at that Link/Port reflecting the updated value resulting from a UIO Write Request to that 64B aligned block implies that all other UIO Requests passing on that Link/Port will observe the UIO Write as fully completed for that 64B aligned block.

The observed sequence is permitted to be different for each 64B aligned block.

A Completer is permitted to implement, through a restricted programming model, an update granularity of less than 64B for some or all of a resource mapped by means of a BAR. In order to maintain the 64B ordering/granularity requirements, such a Completer is permitted to terminate Requests that exceed this smaller update granularity in an implementation specific way.

2.4.4 Update Ordering and Granularity Provided by a Write Transaction §

2.4.4.1 Ordering and Granularity for Non-UIO Writes §

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

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

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

2.4.4.2 Ordering and Granularity for UIO Writes §

For each 64B naturally aligned block updated in-full or in-part by a single UIO Write Request, all bytes updated within the 64B aligned block must be fully observable or not-at-all observable to any read, whether the read is performed by a PCI Express transaction or another mechanism. The observed sequence is permitted to be different for each 64B aligned block but the observed sequence must be consistent for all readers of a particular 64B aligned block.

Once a Completer has Transmitted a UIO Write Completion for a 64B aligned block, the Completer must ensure that all UIO Requests it receives must observe the UIO Write as fully completed for that 64B aligned block. Thus, at any other Link or Port between a Requester and Completer, observation of a UIO Write Completion at that Link/Port implies that all other UIO Requests passing on that Link/Port will observe the UIO Write as fully completed for that 64B aligned block.

The observed sequence is permitted to be different for each 64B aligned block.

A Completer is permitted to implement, through a restricted programming model, an update granularity of less than 64B for some or all of a resource mapped by means of a BAR. In order to maintain the 64B ordering/granularity requirements, such a Completer is permitted to terminate Requests that exceed this smaller update granularity in an implementation specific way.

2.5 Virtual Channel (VC) Mechanism §

The Virtual Channel (VC) mechanism provides support for carrying, throughout the fabric, traffic that is differentiated using $\downarrow\downarrow$ TC $\downarrow\uparrow$ Traffic Class (TC) $\uparrow\downarrow$ labels. The foundations of VCs are independent fabric resources (queues/buffers and associated control logic). These resources are used to move information across Links with fully independent Flow Control between different VCs (Link Flow Control is defined in § Section 2.6.). 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.

As Link speed increases, the buffer space required to support fully independent Flow Control between different VCs while also supporting full Link bandwidth on any given VC also increases. In Flit Mode, the Shared Flow Control (FC) mechanism can be used to reduce this resource requirement. Flow Control is defined in § Section 2.6.

Traffic is associated with VCs by mapping packets with particular TC labels to their corresponding VCs. The Streamlined Virtual $\downarrow\downarrow$ Channel, $\downarrow\uparrow$ Channel (SVC), $\downarrow\uparrow$ Virtual $\downarrow\downarrow$ Channel, $\downarrow\uparrow$ Channel (VC), $\downarrow\uparrow$ and Multi-Function Virtual Channel (MFVC) mechanisms allow flexible mapping of TCs onto the VCs. In the simplest form, TCs can be mapped to VCs on a 1:1 basis. To allow performance/cost tradeoffs, PCI Express provides the capability of mapping multiple TCs onto a single VC. § Section 2.5.2 covers details of TC to VC mapping.

A Virtual Channel is established when one or multiple TCs are associated with a physical VC resource designated by Virtual Channel Identification (VC ID). This process is controlled by configuration software as described in § Section 6.3, § Section 7.9.1, and § Section 7.9.2.

In Flit Mode, initially, VC0 is initialized automatically by hardware with a dedicated FC credit pool, and a shared FC pool. As system software enables other VCs, the enabled VCs are also initialized with a dedicated FC credit pool and a Shared FC pool per VC. The Shared FC credits for additional VCs expand the Shared FC pool available to all VCs (and are permitted to be zero when the appropriate Shared FC credits were granted earlier). When only a single VC is supported and merged credits are not used, there is a single “shared” credit pool and VC0 is initialized with zero dedicated credits. When only a single VC is supported and merged credits are used, there is a single “shared” non-Posted credit pool and VC0 is initialized with zero non-Posted dedicated credits (posted and completion do have dedicated credits).

Once system software has completed enabling all VCs that are to be enabled, it is recommended that system software Set, as appropriate, VC Enablement Completed in the SVC Port Control Register, the All VCs Enabled bit in the Port VC Control Register or the MFVC Port VC Control Register to indicate that VC initialization is completed. Once this bit has been Set, behavior is undefined if additional VCs are enabled or disabled.

The Shared Flow Control Usage Limit mechanism allows system software to manage the allocation of Shared FC by Transmitters, for example to support Quality of Service (QoS) policies.

Support for TCs and VCs beyond the default TC0/VC0 pair is optional although some optional mechanisms also require support for additional TC/VC. The association of TC0 with VC0 is $\downarrow\downarrow$ fixed, i.e., hardwired, $\downarrow\uparrow$ $\downarrow\uparrow$ fixed (i.e., hardwired), $\uparrow\downarrow$ and must be supported by all components. Therefore, the baseline TC/VC setup does not require any VC-specific hardware or software configuration. In order to ensure interoperability where possible, components that do not implement any of the optional SVC, VC, or MFVC Extended Capability structures must obey the following rules:

- A Requester must only generate requests with TC0 label. (Note that if the Requester initiates requests with a TC label other than TC0, the requests may be treated as malformed by the component on the other side of the Link that implements the extended VC Extended Capability and applies TC Filtering.)
- A Completer must accept requests with TC label other than TC0, and must preserve the TC label. That is, any completion that it generates must have the same TC label as the label of the request.

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

Even with the above rules, in some cases interoperability may not be possible, such as when TC/VC mechanisms are used to implement protocols that cannot be mapped onto TC0/VC0. The SVC mechanism and its associated requirements provide a framework for interoperable hardware and software, and it is strongly recommended that hardware implementing support for VCs beyond VC0 support SVC.

A Port containing Functions capable of generating Requests with TC labels other than TC0 must implement suitable SVC, VC, or MFVC Extended Capability structures (as applicable), even if it only supports the default VC. Example Function types are Endpoints and Root Ports. This is required in order to enable mapping of TCs beyond the default configuration. It must follow the TC/VC mapping rules according to the software programming of the ~~SVC, VC, and MFVC Extended Capability structures.~~ extended capability structure used to configure the VC.

SVC provides explicit support for architecturally-defined TC/VC applications via a combination of hardware requirements and software guidance. Ports supporting UIO must implement the SVC Extended Capability. The TC/VC default HW assignments in Table 2-46 are mandatory for SVC.

Table 2-46 Streamlined ~~VC~~ Virtual Channel (SVC) TC/VC Default Assignments

| TC | VC | Description |
|-----|-----|---|
| TC0 | VC0 | TC0 VC0 Default TC/VC- configured automatically by hardware – required to be used for certain mechanisms as defined in this specification |
| TC1 | VC1 | Reserved |
| TC2 | VC2 | Reserved |
| TC3 | VC3 | UIO TC/VC (Required if UIO is supported) |
| TC4 | VC4 | UIO TC/VC (Optional if UIO is supported) |
| TC5 | VC5 | Reserved |
| TC6 | VC6 | Reserved |
| TC7 | VC7 | Reserved |

IMPLEMENTATION NOTE: MULTI-HOST FABRICS AND STREAMLINED VIRTUAL CHANNEL (SVC) TC/VC DEFAULT ASSIGNMENTS §

When PCIe and/or related switching fabrics support multiple hosts, and one or more fabric Links carry traffic from multiple hosts concurrently, the use of TCs across the fabric and TC/VC mappings on each fabric Link may conflict. ~~↑↓E.g., ↓↑For example, ↑~~ if one host relies on TC3 mapping to a UIO VC while another host relies on TC3 mapping to a non-UIO VC, the resulting behavior is undefined.

To avoid TC conflicts on multi-host fabric Links, it is recommended ~~↑↓for↓ ↑↑that↑~~ system software that configures TC/VC ~~↑↓mappings to:↓ ↑↑mappings:↑~~

- ~~↑↓support Streamlined VC (SVC) Extended Capability,↓ ↑↑supports the SVC Extended Capability ,↑~~
- ~~↑↓preserve↓ ↑↑preserves↑ any TC/VC assignments already configured in Switch Ports, whether such VCs are enabled or ↓↑not↓ ↑↑not, and↓~~
- ~~↑↓configure↓ ↑↑configures↑ and ↓↑enable↓ ↑↑enables↑ TC/VC assignments on each RP to match such Switch Ports, to the extent permitted by RP hardware capability.~~

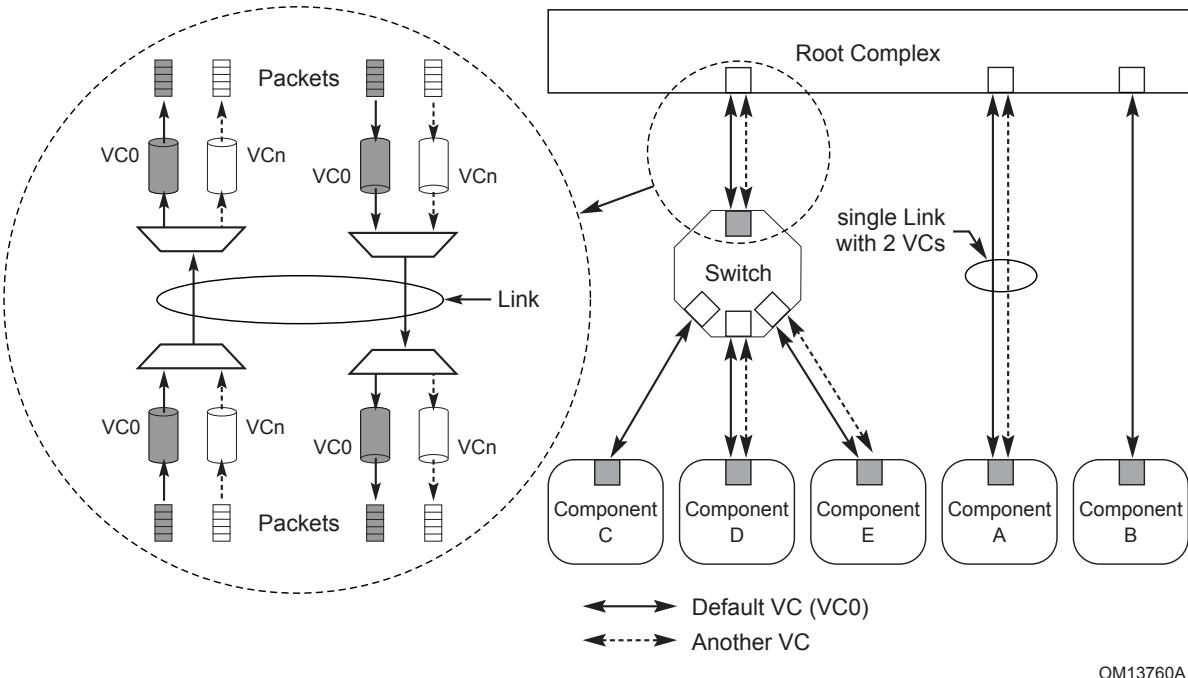
This allows a Fabric Manager to preconfigure TC/VC assignments fabric-wide, and rely on OS cooperation.

For cases where a Fabric Manager does not preconfigure TC/VC assignments, ~~↑↑\$ Table 2-46↑~~ provides reasonable defaults that can work in ~~↑↑currently↑~~ envisioned multi-host fabrics.

System software should use the SVC Extended Capability for TC/VC configuration in all hardware that supports SVC. If it enables VCs other than VC0 in hardware that supports only the VC and/or MFVC Extended Capabilities, it should configure enabled TC/VCs to match the SVC default assignments.

§ Figure 2-94 illustrates the concept of Virtual Channel. Conceptually, traffic that flows through VCs is multiplexed onto a common physical Link resource on the Transmit side and de-multiplexed into separate VC paths on the Receive side.

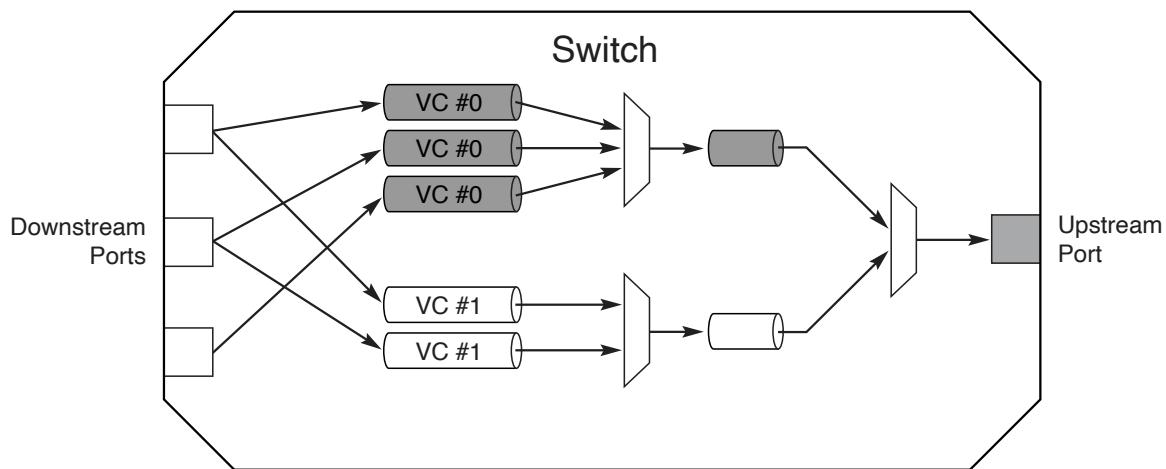
Base 6.4 vs Base 6.3



OM13760A

Figure 2-94 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 2-95 shows conceptually the VC resources within the Switch (shown in § Figure 2-94) that are required to support traffic flow in the Upstream direction.



OM13761

Figure 2-95 Virtual Channel Concept - Switch Internals (Upstream Flow) §

An MFD may implement Virtual Channel resources similar to a subset of those in a Switch, for the purpose of managing the Quality of Service (QoS) for Upstream requests from the different Functions to the device's Upstream Egress Port.

IMPLEMENTATION NOTE: VC AND VC BUFFERING CONSIDERATIONS §

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

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

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

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

2.5.1 Virtual Channel Identification (VC ID) §

PCI Express Ports can support 1 to 8 Virtual Channels - each Port is independently configured/managed therefore allowing implementations to vary the number of VCs supported per Port based on usage model-specific requirements. These VCs are uniquely identified using the VC ID mechanism.

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

Rules for assigning VC ID to VC hardware resources within a Port are as follows:

- VC ID assignment must be unique per Port - The same VC ID cannot be assigned to different VC hardware resources within the same Port.
- VC ID assignment must be the same (matching in the terms of numbers of VCs and their IDs) for the two Ports on both sides of a Link.
- If an MFD implements an MFVC Extended Capability structure, its VC hardware resources are distinct from the VC hardware resources associated with any VC Extended Capability structures of its Functions. The VC ID uniqueness requirement (first bullet above) still applies individually for the MFVC and any VC Extended Capability structures. In addition, the VC ID cross-Link matching requirement (second bullet above) applies for the MFVC Extended Capability structure, but not the VC Extended Capability structures of the Functions.
- VC ID 0 is assigned and fixed to the default VC.
- It is permitted to implement VCs that support only specific protocols and/or use models
 - If software maps such VCs in a way that is incompatible with their protocol/use model requirements, the resulting hardware behavior is undefined

2.5.2 TC to VC Mapping §

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

The mapping of TCs other than TC0 must obey the following rules:

- One or multiple TCs can be mapped to a VC.
- One TC must not be mapped to multiple VCs in any Port or Endpoint Function.
- TC/VC mapping must be identical for Ports on both sides of a Link.
- If UIO is supported, VC3 must be supported, and it must support UIO, and enabling VC3 is required to use UIO.
- If UIO is supported, and if a second UIO VC is supported, then the second UIO VC must be VC4 (and so VC4 must support UIO traffic); if UIO is enabled using only one VC it must be VC3, if UIO is enabled using two VCs they must be VC3 and VC4.
- If a UIO TLP targets an Egress Port where the TC maps to a non-UIO VC, or a non- UIO TLP targets an Egress Port where the TC maps to a UIO VC, such TLPs must be handled as specified in § Section 2.2.1.2 for other TLPs that cannot be translated. Note that this rule partially overlaps with the rule in § Section 2.2.1.2 regarding a UIO TLP targeting an Egress Port in NFM.

§ Table 2-47 provides an example of TC to VC mapping.

Table 2-47 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:

TC_n/VC_k

TC_n mapped to VC_k

TC(_n-_m)/VC_k

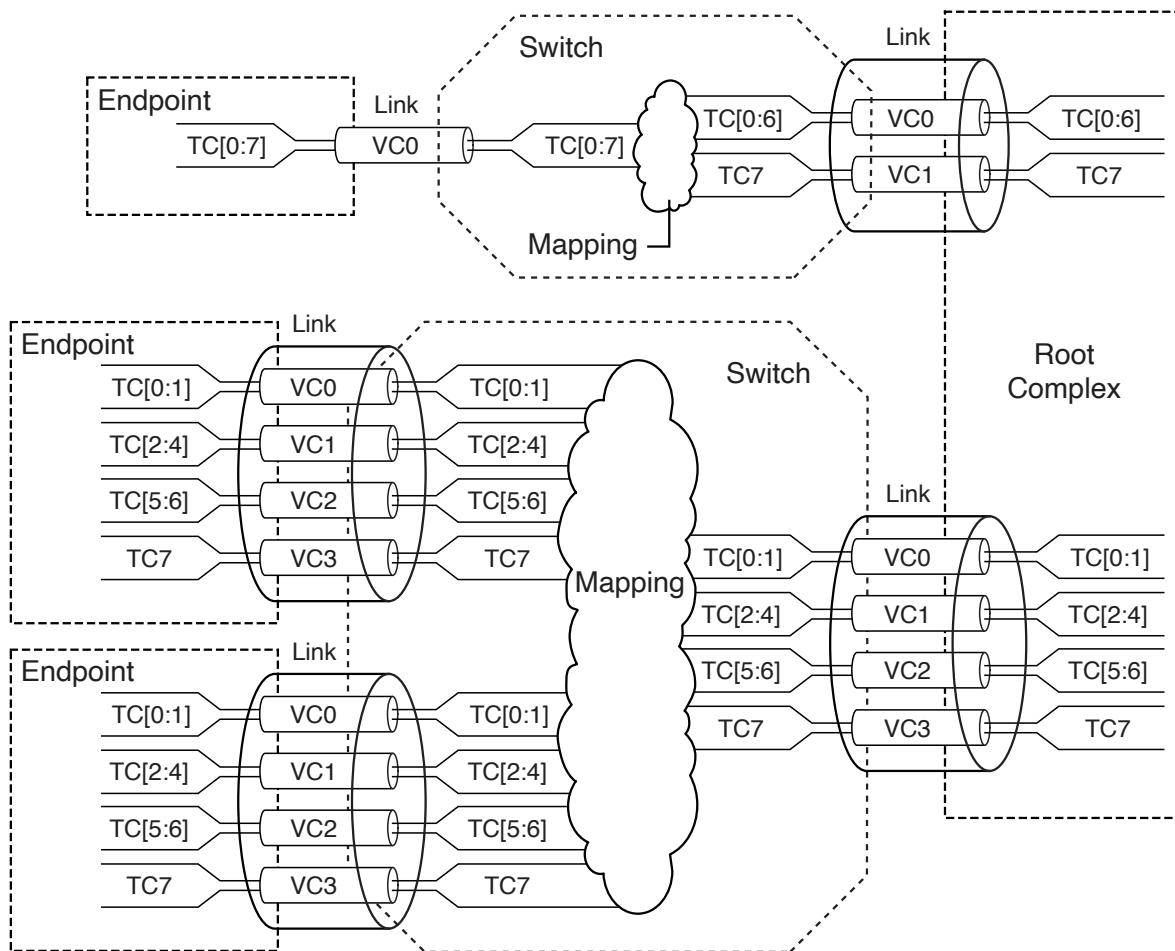
all TCs in the range _n-_m mapped to VC_k (i.e., to the same VC)

TC[_n:_m]/VC[_n:_m]

TC_n/VC_n, TC_{n+1}/VC_{n+1}, ..., TC_m/VC_m

§ Figure 2-96 provides a graphical illustration of TC to VC mapping in several different Link configurations. For additional considerations on TC/VC, refer to § Section 6.3 .

Base 6.4 vs Base 6.3



OM13762

Figure 2-96 An Example of TC/VC Configurations §

2.5.3 VC and TC Rules §

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

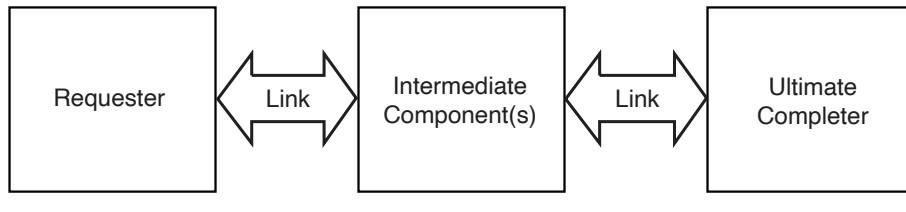
- All devices must support the general purpose I/O Traffic Class (i.e., TC0 and must implement the default VC0).
- Each Virtual Channel (VC) has independent Flow Control.
- There are no ordering relationships required between different TCs.
- There are no ordering relationships required between different VCs.
- A Switch's peer-to-peer capability applies to all Virtual Channels supported by the Switch.
- An MFD's peer-to-peer capability between different Functions applies to all Virtual Channels supported by the MFD.
- Transactions with a TC that is not mapped to any enabled VC in an Ingress Port are treated as Malformed TLPs by the receiving device.
- For Switches, transactions with a TC that is not mapped to any of the enabled VCs in the target Egress Port are treated as Malformed TLPs.

- For a Root Port, transactions with a TC that is not mapped to any of the enabled VCs in the target RCRB are treated as Malformed TLPs.
- For MFDs with an MFVC Extended Capability structure, any transaction with a TC that is not mapped to an enabled VC in the MFVC Extended Capability structure is treated as a Malformed TLP.
- Switches must support independent TC/VC mapping configuration for each Port.
- A Root Complex must support independent TC/VC mapping configuration for each RCRB, the associated Root Ports, and any RCiEPs.

For more details on the VC and TC mechanisms, including configuration, mapping, and arbitration, refer to § [Section 6.3](#).

2.6 Ordering and Receive Buffer Flow Control §

Flow Control (FC) is used to prevent overflow of Receiver buffers and to enable compliance with the ordering rules defined in § [Section 2.4](#). Note that the Flow Control mechanism is used by the Requester to track the queue/buffer space available in the agent across the Link as shown in § [Figure 2-97](#). That is, Flow Control is point-to-point (across a Link) and not end-to-end. Flow Control does not imply that a Request has reached its ultimate Completer.



OM13776

Figure 2-97 Relationship Between Requester and Ultimate Completer §

Flow Control is orthogonal to the data integrity mechanisms used to implement reliable information exchange between Transmitter and Receiver. Flow Control can treat the flow of TLP information from Transmitter to Receiver as perfect, since the data integrity mechanisms ensure that corrupted and lost TLPs are corrected through retransmission (see § [Section 3.6](#)).

In Non-Flit Mode each Virtual Channel (VC) maintains an independent FC credit pool.

In Flit Mode, the Shared FC mechanism can be used to reduce VC resource requirements. There are two sets of resources associated with each VC: a (typically small) pool of dedicated resources associated independently with each FC/VC (to avoid deadlock by allowing that the Transmitter to transmit at least one TLP in that VC/FC using only dedicated credit(s)), and a portion of the (typically larger) pool of shared resources. The Transmitter gate function (defined later in this section) uses the sum of all Shared FC returned across all VCs. The transmitter gate function also provides a Usage Limit mechanism to avoid over-consumption of buffers by stalled VCs. This Usage Limit mechanism is configured by software and defaults to disabled. To support Usage Limit, credits are returned to the Transmitter indicating the VC for the TLP(s) that, by making forward progress, freed those credits. The FC information is conveyed between two sides of the Link using DLLPs. The VC ID field of the DLLP is used to carry the VC ID that is required for proper Flow Control credit accounting. Additionally, [Merged] FC enables the sharing of buffers for Posted Requests and Completions, further reducing resource requirements.

Flow Control mechanisms used internally within an MFD are outside the scope of this specification.

Flow Control is handled by the Transaction Layer in cooperation with the Data Link Layer. The Transaction Layer performs Flow Control accounting functions for Received TLPs and “gates” TLP Transmissions based on available credits for transmission even if those TLPs are eventually nullified.

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

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

2.6.1 Flow Control (FC) Rules §

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

- Flow Control (FC) information is transferred using Flow Control Packets (FCPs), which are a type of DLLP (see § Section 3.5), and, in some cases, the Optimized_Update_FC.
- **FC Unit Size** indicates the number of DW covered by one flow control credit:
 - For Data, the FC Unit Size is 4 DW.
 - For Headers in Non-Flit Mode:
 - For Receivers that do not support TLP Prefixes, FC Unit Size is the sum of one maximum-size Header and TLP Digest.
 - For Receivers that support End-End TLP Prefixes, FC Unit Size is the sum of one maximum-size Header, TLP Digest, and the maximum number of End-End TLP Prefixes permitted in a TLP.
 - The management of FC for Receivers that support Local TLP Prefixes is dependent on the Local TLP Prefix type.
 - For Headers in Flit Mode:
 - For Switch Port Receivers, FC Unit Size is the sum of one maximum-size Base Header, OHC-A, OHC-B, OHC-C, OHC-E if supported, and one maximum-size TLP Trailer.
 - For Endpoint Upstream Port and Root Port Receivers, FC Unit Size is the sum of one Base Header of the largest supported size, OHC-A, OHC-B, OHC-C, OHC-E if supported, and one TLP Trailer of the largest size supported.
- For NFM and for dedicated credits in FM, each Virtual Channel has independent FC.
- In FM, each Virtual Channel has some amount of independent FC referred to as dedicated credits, and some amount of shared FC.
 - When only one single VC is implemented and [Merged] is not used, there are no dedicated credits, all flow control uses shared credits (see § Table 3-3, Notes 3 and 4).
- It is permitted for a Transmitter to use dedicated credits when Transmitting a TLP, even when sufficient shared credits are available.
- The Transmitter indicates the use of dedicated credits for a specific TLP by applying the Flit Mode Local TLP Prefix with the TLP Uses Dedicated Credits bit Set.
- Flow Control distinguishes three types of TLPs (note relationship to ordering rules - see § Section 2.4):
 - Posted Requests (P) - Messages and Memory Writes
 - Non-Posted Requests (NP) - All Reads, I/O Writes, Configuration Writes, AtomicOps, and DMWr.
 - 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 2-48.

Table 2-48 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 2-49.

Table 2-49 TLP Flow Control Credit Consumption §

| TLP | Credit Consumed ⁵³ |
|--|--|
| Memory, I/O, Configuration Read Request | 1 NPH unit |
| Memory Write Request | 1 PH + n PD units ⁵⁴ |
| I/O, Configuration Write Request | 1 NPH + 1 NPD Note: size of data written is never more than 1 (aligned) DW |
| AtomicOp, DMWr Request | 1 NPH + n NPD units |
| Message Requests without data | 1 PH unit |
| Message Requests with data | 1 PH + n PD units |
| Memory Read Completion | 1 CplH + n CplD units |
| I/O, Configuration Read Completions | 1 CplH unit + 1 CplD unit |
| I/O, Configuration Write, and DMWr Completions | 1 CplH unit |
| AtomicOp Completion | 1 CplH unit + 1 CplD unit Note: size of data returned is never more than 4 (aligned) DWs. |

- FC must be initialized autonomously by hardware only for the default Virtual Channel (VC0).

53. Each header credit implies the ability to accept a TLP Digest along with the corresponding TLP.

54. For all cases where “n” appears, n = Roundup(Length/ FC unit size). Where Length is the size of the Payload in DW.

- VC0 is initialized when the Data Link Layer is in the DL_Init state following reset (see § Section 3.2 and § Section 3.4).
- When Virtual Channels other than VC0 are enabled by software, each newly enabled VC must follow the Flow Control initialization protocol (see § Section 3.4).
 - Software enables a Virtual Channel by setting the VC Enable bits for that Virtual Channel in both components on a Link (see § Section 7.9.1 and § Section 7.9.2).

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

- Software disables a Virtual Channel by clearing the VC Enable bits for that Virtual Channel in both components on a Link.
 - Disabling a Virtual Channel for a component resets the Flow Control tracking mechanisms for that Virtual Channel in that component.
 - In Flit Mode, disabling a Virtual Channel resets the Flow Control tracking mechanisms for dedicated credits for that Virtual Channel in that component and has no effect on Shared Flow Control credit tracking.
 - In Flit Mode, behavior is undefined if a VC is disabled and subsequently re-enabled while the link remains up.
- InitFC1 and InitFC2 FCPs are used only for Flow Control initialization (see § Section 3.4).
- An InitFC1, InitFC2, UpdateFC FCP, or Optimized_Update_FC that specifies a Virtual Channel that is disabled must be 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 2-50.
 - Scaled Flow Control is activated when both Ports on a Link perform the Data Link Feature mechanism with the Scaled Flow Control Supported bit Set (i.e., Local Scaled Flow Control Supported and Remote Scaled Flow Control Supported are both Set, see § Section 3.3).
 - If Scaled Flow Control is not supported or supported but not activated, use the values in the "Scale Factor 1" column.
- If Scaled Flow Control is supported and activated, use the values in the column for the scaling factor associated with that credit type (see § Section 3.4.2).
- For a Multi-Function Device where different Functions have different Rx_MPS_Limit values, the largest Rx_MPS_Limit value across all Functions must be used.
- In Flit Mode, for each Credit Type, shared credit advertisement during initialization must either:
 - Advertise [Infinite.3] on all VCs, or
 - Advertise a combination of [Zero] and non- [Zero] on all VCs.
 - When multiple VCs are supported, it is permitted for all VCs to advertise [Zero] shared credits.
 - All VCs advertising non- [Zero] shared credits must have the same Scale Factor.
 - If any VC advertised non- [Zero] shared credits:
 - All VCs advertising [Zero] shared credits must use that Scale Factor in subsequent UpdateFCs.
 - The sum of the advertisements across all VCs must be greater than or equal to the value in § Table 2-50 multiplied by the number of enabled VCs. The § Table 2-50 minimum values do not apply to an individual VC as long as this rule applies.

Base 6.4 vs Base 6.3

- When [Merged] flow control is used, the sum of the Posted advertisements across all VCs must be greater than or equal to the value in § Table 2-50 multiplied by two times the number of enabled VCs (i.e., Posted minimum must account for Completions as well).
- See § Table 3-3 Note 6.

- In Flit Mode, dedicated credits are permitted to use any Scale Factor on any VC for any Credit Type.

Table 2-50 Minimum Initial Flow Control Advertisements⁵⁵ §

| Credit Type | Minimum Advertisement | | |
|-------------|---|--|--|
| | No Scaling or Scale Factor 1 | Scale Factor 4 | Scale Factor 16 |
| PH | Shared credits in Flit Mode: 4 units – credit value of 04h. Otherwise: 1 unit - credit value of 01h. | 4 Units - credit value of 01h. | 16 Units - credit value of 01h. |
| PD | Shared credits in Flit Mode: $\text{Ceiling}(\text{Rx_MPS_Limit} / (\text{FC Unit Size} * 4)) + 4$. Otherwise: Rx_MPS_Limit divided by FC Unit Size . Example: If the Rx_MPS_Limit is 1024 bytes, the smallest permitted initial credit value would be 040h (44h for shared credits in Flit Mode). | $\text{Ceiling}(\text{Rx_MPS_Limit} / (\text{FC Unit Size} * 4)) + 1$. Example: If the Rx_MPS_Limit is 1024 bytes, the smallest permitted initial credit value would be 011h. | $\text{Ceiling}(\text{Rx_MPS_Limit} / (\text{FC Unit Size} * 16)) + 1$. Example: If the Rx_MPS_Limit is 1024 bytes, the smallest permitted initial credit value would be 005h. |
| NPH | Shared credits in Flit Mode: 4 units – credit value of 04h. Otherwise: 1 unit - credit value of 01h. | 4 Units - credit value of 01h. | 16 Units - credit value of 01h. |
| NPD | Shared credits in Flit Mode: $\text{Max}(\text{Rx_NP_MPS_Limit} / \text{FC Unit Size}, 4) + 4$ (Note 3) Otherwise: Rx_NP_MPS_Limit divided by FC Unit Size (Note 3) | $\text{Ceiling}(\text{Rx_NP_MPS_Limit} / (\text{FC Unit Size} * 4)) + 1$ (Note 3) | $\text{Ceiling}(\text{Rx_NP_MPS_Limit} / (\text{FC Unit Size} * 16)) + 1$ (Note 3) |
| CplH | Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: for shared credits in Flit Mode: 4 units – credit value of 04h, otherwise 1 FC unit - credit value of 01h Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units (Note 1). In Flit Mode, if [Merged] is enabled (see § Section 3.4.1), PH credits are used for Completions in the shared pool. | Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: 4 FC units - credit value of 01h Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units (Note 1). In Flit Mode, if [Merged] is enabled (see § Section 3.4.1), PH credits are used for Completions in the shared pool. | Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: 16 FC units - credit value of 01h Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units (Note 1). In Flit Mode, if [Merged] is enabled (see § Section 3.4.1), PH credits are used for Completions in the shared pool |

55. PCI Express to PCI/PCI-X Bridge requirements are addressed in [PCIe-to-PCI-PCI-X-Bridge].

| Credit Type | Minimum Advertisement | | |
|-------------|---|---|--|
| | No Scaling or Scale Factor 1 | Scale Factor 4 | Scale Factor 16 |
| CplD | Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: for shared credits in Flit Mode $\text{Max}(\text{Rx_MPS_Limit} / \text{FC Unit Size}, 4) + 4$ (Note 3), otherwise Rx_MPS_Limit divided by FC Unit Size. | Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: Ceiling($\text{Rx_MPS_Limit} / (\text{FC Unit Size} * 4)$) + 1. | Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: Ceiling($\text{Rx_MPS_Limit} / (\text{FC Unit Size} * 16)$) + 1. |
| | Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units (Note 2). | Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units (Note 2). | Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units (Note 2). |
| | In Flit Mode, if [Merged] is enabled (see § Section 3.4.1), PD credits are used for Completions in the shared pool. | In Flit Mode, if [Merged] is enabled (see § Section 3.4.1), PD credits are used for Completions in the shared pool. | In Flit Mode, if [Merged] is enabled (see § Section 3.4.1), PD credits are used for Completions in the shared pool. |

Notes:

1. Infinite header credits is an encoding that is interpreted as infinite by the Transmitter, which will, therefore, never throttle. In Flit Mode the [Infinite.3] encoding is used (see § Table 3-3). In Non-Flit Mode the [Infinite.1] or [Infinite.2] encodings are used (see § Table 3-2).
2. Infinite data credits is an encoding that is interpreted as infinite by the Transmitter, which will, therefore, never throttle. In Flit Mode the [Infinite.3] encoding is used (see § Table 3-3). In Non-Flit Mode the [Infinite.1] or [Infinite.2] encodings are used (see § Table 3-2).
3. **Rx_NP_MPS_Limit** is the maximum size Non-Posted TLP Payload accepted by the Receiver (in DW). Larger of:
 - Payload size supported by any implemented earmarked TLP Type values.
 - Receiver that supports DMWr routing capability or DMWr Completer capability:
 - If DMWr Request Routing Supported is 1: 128 bytes (8 credit units)
 - If DMWr Request Routing Supported is 0 and DMWr Completer Supported is 1 and DMWr Lengths Supported is 00b: 64 bytes (4 credit units)
 - If DMWr Request Routing Supported is 0 and DMWr Completer Supported is 1 and DMWr Lengths Supported is not 00b: 128 bytes (8 credit units)
 - Receiver that supports AtomicOp routing capability or any AtomicOp Completer capability:
 - If AtomicOp Routing Supported is 1: 32 bytes (2 credit units)
 - If AtomicOp Routing Supported is 0 and 128-bit CAS Completer Supported is 0: 16 bytes (1 credit unit)
 - If AtomicOp Routing Supported is 0 and 128-bit CAS Completer Supported is 1: 32 bytes (2 credit units)
 - 16 bytes (1 credit unit)

- A Root Complex that supports no peer-to-peer traffic between Root Ports must advertise infinite Completion credits on every Root Port.
- A Root Complex that supports peer-to-peer traffic between some or all of its Root Ports may optionally advertise non-infinite Completion credits on those Root Ports. In this case, the Root Complex must ensure that deadlocks are avoided and forward progress is maintained for completions directed towards the Root Complex. Note that temporary stalls of completion traffic (due to a temporary lack of credit) are possible since Non-Posted **↓↑requests↓↑Requests↑** forwarded by the RC may not have explicitly allocated completion buffer space.

Base 6.4 vs Base 6.3

- A Receiver that does not support Scaled Flow Control must never cumulatively issue more than 2047 outstanding unused credits to the Transmitter for data or 127 for header. A Receiver that supports Scaled Flow Control must never cumulatively issue more outstanding unused data or header credits to the Transmitter than the Max Credits values shown in § Table 3-4 .
 - Components may optionally check for violations of this rule. If a component implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE).
 - If checked, this is a reported error associated with the Receiving Port (see § Section 6.2)
- If [Infinite.1] , [Infinite.2] , or [Infinite.3] credit advertisement has been made during initialization, no Flow Control updates are required following initialization.
 - If UpdateFC DLLPs or Optimized_Update_FCs are sent, the credit value fields must be Clear and must be ignored by the Receiver. The Receiver may optionally check for non-zero update values (in violation of this rule). If a component implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE)
 - If checked, this is a reported error associated with the Receiving Port (see § Section 6.2)
- If Scaled Flow Control is activated, the HdrScale and DataScale fields in the ~~↑↓UpdateFCs↓~~ ~~↑↑UpdateFC DLLPs↑~~ must match the values advertised during initialization (see § Section 3.4.2) with the following exceptions.
 - In Flit Mode, when more than one VC is supported, it is permitted to advertise [Zero] shared credits during initialization. For VCs that initialized with [Zero] shared credits, the HdrScale and DataScale fields in shared credit ~~↑↓UpdateFCs↓~~ ~~↑↑UpdateFC DLLPs↑~~ must match the non- [Zero] HdrScale and DataScale values used by other VCs. If [Zero] shared credits were advertised on all VCs, the HdrScale and DataScale fields in the corresponding shared credit ~~↑↓UpdateFCs↓~~ ~~↑↑UpdateFC DLLPs↑~~ are undefined.
 - In Flit Mode, it is permitted to advertise [Merged] shared completion credits during initialization. In this situation, the HdrScale and DataScale fields in shared completion credit ~~↑↓UpdateFCs↓~~ ~~↑↑UpdateFC DLLPs↑~~ must match the values advertised for shared posted credits during initialization. When [Merged] shared completion credits are advertised, at least one VC must ~~↑↓advertize↓~~ ~~↑↑advertise↑~~ non- [Zero] shared posted completion credits.
 - The Receiver may optionally check for violations of this rule. If a Receiver implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE).
 - If checked, this is a reported error associated with the Receiving Port (see § Section 6.2).
- A received TLP using a VC that is not enabled is a Malformed TLP.
 - VC0 is always enabled.
 - For VCs 1-7, a VC is considered enabled when the corresponding VC Enable ~~↑↑/ SVC Enable↑~~ bit ~~↑↓in the VC Resource Control register↓~~ has been ~~↑↓Set,↓~~ ~~↑↑Set↑~~ and ~~↑↓once↓~~ FC negotiation for that VC has exited the FC_INIT1 state ~~↑↓and progressed↓~~ to the FC_INIT2 state (see § Section 3.4).
 - This is a reported error associated with the Receiving Port (see § Section 6.2).
- TLP transmission using any VC 0-7 is not permitted until initialization for that VC has completed by exiting FC_INIT2 state.

For VCs 1-7, software must use the ~~↑↑corresponding↑~~ VC Negotiation Pending ~~↑↑/ SVC Negotiation Pending↑~~ bit in the corresponding VC Resource Status Register to ensure that a VC is not used until negotiation has completed by exiting the FC_INIT2 state in both components on a Link.

In Flit Mode, if software disables VC 1-7:

- Dedicated Credit counts are cleared.
- Shared credit counts are not affected. Shared credits are returned as usual when the associated TLPs are consumed by the Receiver and are available for use by the remaining VCs.

- Behavior is undefined if software subsequently re-enables VC1-7.

The **[Field Size]** parameter used in the following sections is described in § Table 2-51 (see § Section 3.4.2 for details of Scaled Flow Control).

Table 2-51 [Field Size] Values §

| Scaled Flow Control Supported | HdrScale or DataScale | [Field Size] for PH, NPH, CplH | [Field Size] for PD, NPD, CplD |
|-------------------------------|-----------------------|--------------------------------|--------------------------------|
| No | x | 8 | 12 |
| Yes | 00b | 8 | 12 |
| Yes | 01b | 8 | 12 |
| Yes | 10b | 10 | 14 |
| Yes | 11b | 12 | 16 |

In Flit Mode, the following rules apply to **[Merged]** FC credits:

- Receivers are permitted to support **[Merged]** FC; Transmitters must support **[Merged]** FC.
- When **[Merged]** FC is enabled:
 - Shared Completion Header credits are **[Merged]** with shared Posted Header credits.
 - During FC initialization, shared Posted Header credits must be used to indicate the total **[Merged]** shared Header credit pool.
 - FC updates must indicate either shared Posted Header credits or shared Completion Header credits according to the type of credit freed by the Receiver.
 - Shared Completion Data credits are **[Merged]** with shared Posted Data credits.
 - During FC initialization, shared Posted Data credits must be used to indicate the total **[Merged]** shared Data credit pool.
 - FC updates must indicate either shared Posted Data credits or shared Completion Data credits according to the type of credit freed by the Receiver.
- Dedicated Header credits must not be **[Merged]**.
- Dedicated Data credits must not be **[Merged]**.
- Merging behavior for each link direction is independent. The Receivers on each end of a given Link choose whether or not to merge independently
- Use of **[Merged]** shared credits must be consistent across VCs. If one VC uses **[Merged]** shared credits, all VCs must also use **[Merged]** shared credits.
- Use of **[Merged]** shared credits must be consistent between Hdr and Data. If Hdr uses **[Merged]** shared credits, Data must also use **[Merged]** shared credits.
- When **[Merged]** FC is enabled, it must be ensured that a Requester's rate of Completion processing be matched to that Requester's rate of issuing the corresponding Requests as measured within a sliding window of not more than 100 µs.

In Flit Mode, the Receiver must return credits indicating the VC of the buffer(s) freed. If **[Merged]** FC is enabled, the Receiver must return credits indicating the FC Type of the buffer(s) freed.

When more than one VC advertises shared credits with scale factor 01b, 10b, or 11b, that scale factor must be able to express all allocated shared credits, regardless of VC. For example, if VC0 and VC1 each advertise 120 header credits (i.e.,

a total of 240 credits), they must do so using a scale factor other than 1 (01b) since that scale factor is limited to 127 outstanding credits.

In Flit Mode, shared credits for Header and Data are managed in credit blocks, where a credit block consists of 4 credits of the appropriate type. Credit blocks are not affected by the scale factor. Credit blocks do not apply to dedicated credits.

Rules for FC accounting with credit blocks:

- In each VC, per FC Type, shared credits must be reserved by the Transmitter and released by the Receiver in units of credit blocks.
- When a single TLP does not fully consume all the credits in a credit block, the remaining credits in the credit block must be allocated for consumption only by TLP(s) in the same VC and of the same FC Type.
 - Credit block allocation must distinguish between Posted and Completion FC Types, regardless of whether [Merged] credits are used for FC accounting.
 - Once a credit block is allocated, it must be held open until fully consumed by TLP(s) in the same VC and of the same FC Type, or until the associated VC is disabled and/or a DL_Down condition is entered.
 - Once a credit block is allocated, it must be applied only for TLP(s) in the same VC and of the same FC Type, even if TLP(s) in other VCs and/or of other FC Types are Transmitted/Received.
- Receivers must advertise credits in units of whole credit blocks.

If Shared ~~↓↓credits↓~~ ~~↑↑Credits↑~~ are infinite for a given FC/VC, Shared and Dedicated ~~↓↓credits↓~~ ~~↑↑Credits↑~~ in all VCs for that FC must be infinite. For Example, if VC0 and VC1 are enabled and Shared Completion Header credits are infinite in VC0:

- Dedicated Completion Header credits in VC0 must be infinite.
- Dedicated Completion Header credits in VC1 must be infinite.
- Shared Completion Header credits in VC1 must be infinite.

IMPLEMENTATION NOTE: MOTIVATION FOR SHARED CREDIT BLOCKS §

Shared FC enables the reduced cost implementation of multiple Virtual Channels by allowing common sets of resources to be shared. However, cost and complexity are increased, relative to the use of only dedicated credits, by the need to track TLPs as they are stored and removed from these shared structures. Because TLPs in different VCs are unordered with respect to each other, and often have different traffic behaviors, shared resource tracking typically requires linked-list structures for tracking TLPs in each VC, and TLP storage quickly becomes fragmented.

If, as is typically the case, the TLP storage is block-oriented, then the additional constraints of efficient block management motivate that TLPs in a given VC can be packed together. However, it is very complex to manage sub-block level out-of-order removal of TLPs in different VCs and the subsequent re-allocation of that freed space. So, by requiring both the Transmitter and Receiver to explicitly recognize credit blocks, the Receiver's buffer management logic is considerably simplified, while maintaining the efficient use of Receiver resources.

§ Figure 2-98 illustrates how a series of received TLPs would be placed into the Receiver's buffers.

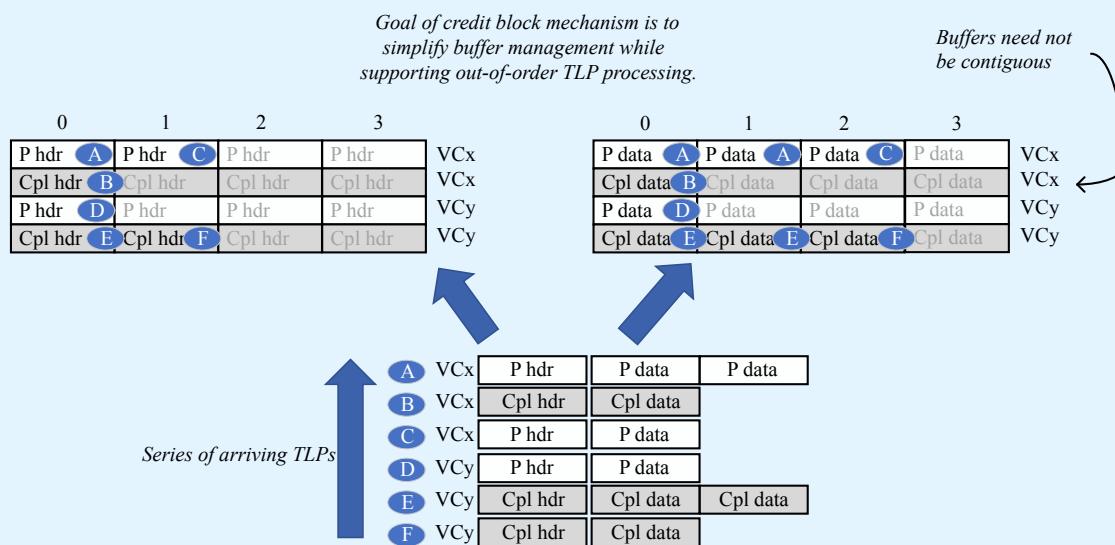


Figure 2-98 Credit Block Example

2.6.1.1 FC Information Tracked by Transmitter

- For each type of information tracked, there are two quantities (Non-Flit Mode) or six quantities (Flit Mode) tracked for Flow Control TLP Transmission gating:
 - **CREDITS_CONSUMED** (per VC, all modes)
 - In Non-Flit Mode, CREDITS_CONSUMED is updated for all TLPs.
 - In Flit Mode, CREDITS_CONSUMED is updated for TLPs transmitted using dedicated credits.

- Count of the total number of FC units consumed by TLP Transmissions made since Flow Control initialization, modulo $2^{[\text{Field Size}]}$ (where Field Size is defined in § Table 2-51).
- Set to all 0's at interface initialization
- Set to 0 when VC Enable for VC[i] is Cleared.
- Updated for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission as shown:

$$\underline{\text{CREDITS_CONSUMED}} := (\underline{\text{CREDITS_CONSUMED}} + \underline{\text{Increment}}) \bmod 2^{[\text{Field Size}]}$$

Equation 2-1 CREDITS_CONSUMED §

(Where *Increment* is the size in FC credits of the corresponding part of the TLP passed through the gate, and Field Size is defined in § Table 2-51)

- **SHARED_CREDITS_CONSUMED** (per VC, Flit Mode only)
 - In Non-Flit Mode, SHARED_CREDITS_CONSUMED is not used.
 - In Flit Mode, SHARED_CREDITS_CONSUMED is updated for TLPs transmitted using shared credits.
 - Count of the total number of FC units consumed by TLP Transmissions made since Flow Control initialization, modulo $2^{[\text{Field Size}]}$ (where Field Size is defined in § Table 2-51).
 - Set to all 0's at interface initialization
 - Updated for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission using shared credits as shown:

$$\underline{\text{SHARED_CREDITS_CONSUMED}} := (\underline{\text{SHARED_CREDITS_CONSUMED}} + \underline{\text{Increment}}) \bmod 2^{[\text{Field Size}]}$$

Equation 2-2 SHARED_CREDITS_CONSUMED §

(Where *Increment* is the size in FC credits of the corresponding part of the TLP passed through the gate, and Field Size is defined in § Table 2-51)

- SHARED_CREDITS_CONSUMED is 0 for VCs that are not implemented or have never been enabled.
- SHARED_CREDITS_CONSUMED is preserved when VC Enable for VC 1-7 is Cleared.
- SHARED_CREDITS_CONSUMED is maintained independently for Posted and Completion credits even when Merged was selected by the Receiver.
- **SUM_SHARED_CREDITS_CONSUMED** (per Port, Flit Mode only)

$$\text{SUM_SHARED_CREDITS_CONSUMED} = \left(\sum_{i=0}^{i=7} \text{SHARED_CREDITS_CONSUMED}[i] \right) \bmod 2^{\text{[Field Size]}}$$

Equation 2-3 SUM_SHARED_CREDITS_CONSUMED §

- **SHARED_CREDITS_CONSUMED_CURRENTLY** (per VC, Flit Mode only, abbreviated as **SCCC** below)
 - In Non-Flit Mode, SHARED_CREDITS_CONSUMED_CURRENTLY is not used.
 - In Flit Mode, SHARED_CREDITS_CONSUMED_CURRENTLY is updated for TLPs transmitted using shared credits and for ~~↓↓UpdateFCs↓~~ ↓↑UpdateFC DLLPs↓ that return shared credits.
 - Set to all 0's at interface initialization
 - Updated for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission using shared credits as shown:

$$\underline{\text{SCCC}}[i] := (\underline{\text{SCCC}}[i] + \text{Increment}) \bmod 2^{\text{[Field Size+1]}}$$

Equation 2-4 TLP SHARED_CREDITS_CONSUMED_CURRENTLY §

(Where *Increment* is the size in FC credits of the corresponding part of the TLP passed through the gate, and [Field Size] is defined in § Table 2-51)

- Updated for each UpdateFC releasing shared credit from VC[i]:

$$\underline{\text{SCCC}}[i] := \left(\underline{\text{SCCC}}[i] - (\text{UpdateFC value} - \underline{\text{SHARED_CREDIT_LIMIT}}[i]) \bmod 2^{\text{[Field Size]}} \right) \bmod 2^{\text{[Field Size+1]}}$$

Equation 2-5 FC SHARED_CREDITS_CONSUMED_CURRENTLY §

- SHARED_CREDITS_CONSUMED_CURRENTLY is preserved when VC Enable for VC 1-7 is Cleared.
- SHARED_CREDITS_CONSUMED_CURRENTLY is maintained independently for Posted and Completion credits even when [Merged] was selected by the Receiver.
- **TOTAL_SHARED_CREDITS_AVAILABLE** (per Port, Flit Mode only)
 - In Non-Flit Mode, TOTAL_SHARED_CREDITS_AVAILABLE is not used.
 - In Flit Mode, TOTAL_SHARED_CREDITS_AVAILABLE contains the sum of the shared credits granted for all VCs during flow control initialization.
 - For [Merged], this initial value for all shared Completion credits is 0.
 - For [Zero], this initial value for that VC is 0.
 - TOTAL_SHARED_CREDITS_AVAILABLE is not affected when VC Enable for VC 1-7 is Cleared.
- **CREDIT_LIMIT** (per VC, all modes)

Base 6.4 vs Base 6.3

- In Non-Flit Mode, CREDIT_LIMIT reflects all credit flow control updates.
 - In Flit Mode, CREDIT_LIMIT reflects dedicated credit flow control updates.
 - CREDIT_LIMIT contains the most recent number of FC units legally advertised by the Receiver. This quantity represents the total number of FC credits made available by the Receiver since Flow Control initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is defined in § Table 2-51).
 - Undefined at interface initialization
 - Set to the value indicated during Flow Control initialization
 - For “infinite” credits, this value is 0
 - For each FC update received,
 - $\uparrow\downarrow$ In Non-Flit Mode, $\uparrow\downarrow$ if CREDIT_LIMIT is not equal to the update value, set CREDIT_LIMIT to the update $\uparrow\downarrow$ value $\downarrow\uparrow$
 - $\uparrow\downarrow$ In Flit Mode, if Dedicated Flow Control is indicated (see § Figure 3-11) and CREDIT_LIMIT is not equal to the update value, set CREDIT_LIMIT to the update $\uparrow\downarrow$ value $\downarrow\uparrow$
- Errata: Base 6.3
B842 Δ \triangleleft
- **SHARED_CREDIT_LIMIT** (per VC, Flit Mode only)
 - In Non-Flit Mode, SHARED_CREDIT_LIMIT is not used.
 - In Flit Mode, SHARED_CREDIT_LIMIT reflects shared credit flow control updates.
 - SHARED_CREDIT_LIMIT contains the most recent number of FC units legally advertised by the Receiver. This quantity represents the total number of shared FC credits made available by the Receiver since Flow Control initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is defined in § Table 2-51).
 - Undefined at interface initialization
 - Set to the value indicated during initial Flow Control initialization.
 - For Merged, this initial value for all shared Completion credits is 0.
 - For Zero, this initial value is 0.
 - SHARED_CREDIT_LIMIT is preserved when VC Enable for VC 1-7 is Cleared and also preserved during subsequent Flow Control initialization.
 - For each FC update received,
 - $\uparrow\downarrow$ In Flit Mode, if Shared Flow Control is indicated (see § Figure 3-11) and $\uparrow\downarrow$ if SHARED_CREDIT_LIMIT is not equal to the update value, set SHARED_CREDIT_LIMIT to the update value
- Errata: Base 6.3
B842 Δ \triangleleft
- **SUM_SHARED_CREDIT_LIMIT** (per Port, $\uparrow\downarrow$ flit $\downarrow\uparrow$ Flit $\uparrow\downarrow$ Mode only)
 - In Non-Flit Mode, SUM_SHARED_CREDIT_LIMIT is not used.
 - In Flit Mode, SUM_SHARED_CREDIT_LIMIT is defined by § Equation 2-6 .

$$\text{SUM_SHARED_CREDIT_LIMIT} = \left(\sum_{i=0}^{i=7} \text{SHARED_CREDIT_LIMIT}[i] \right) \bmod 2^{\text{[Field Size]}}$$

Equation 2-6 SUM_SHARED_CREDIT_LIMIT §

This equation is not affected by [Merged]. When [Merged] is requested by the Receiver, Posted and Completion credits have distinct versions of SUM_SHARED_CREDIT_LIMIT.

- If a Transmitter detects that a TLP it is preparing to transmit is malformed, the Transmitter *MUST@FLIT* discard the TLP and handle the condition as an Uncorrectable Internal Error.
- If a Transmitter detects that a TLP it is preparing to transmit appears to be properly formed but with bad ECRC, the Transmitter *MUST@FLIT* transmit the TLP and update its internal Flow Control credits accordingly.
- The Transmitter gating function must determine if sufficient credits have been advertised to permit the transmission of a given TLP. If the Transmitter does not have enough credits to transmit the TLP, it must block the transmission of the TLP, possibly stalling other TLPs that are using the same Virtual Channel. The Transmitter must follow the ordering and deadlock avoidance rules specified in § Section 2.4, which require that certain types of TLPs must bypass other specific types of TLPs when the latter are blocked. Note that TLPs using different Virtual Channels have no ordering relationship and must not block each other.
- In Flit Mode, the **shared transmitter gating function** test is performed as follows:
 - Credits must be allocated to specific VCs per the credit block rules defined in § Section 2.6.1.
 - For each required type of credit, the number of credits required is calculated as:

$$\text{SHARED_CUMULATIVE_CREDITS_REQUIRED} = (\text{SUM_SHARED_CREDITS_CONSUMED} + \text{credit units required for pending TLP}) \bmod 2^{\text{[Field Size]}}$$

Equation 2-7 SHARED_CUMULATIVE_CREDITS_REQUIRED §

This equation is not affected by [Merged]. When [Merged] is requested by the Receiver, Posted and Completion credits have distinct versions of SHARED_CUMULATIVE_CREDITS_REQUIRED.

- The transmitter is permitted to transmit a TLP if any of the following are true:
 - SHARED_CREDIT_LIMIT was “infinite” during Flow Control initialization.
 - For Non-Posted credits and for Posted and Completion credits when [Merged] was not requested by the Receiver, Shared Flow Control Usage Limit Enable is Clear and, for each type of information in the TLP, § Equation 2-8 is satisfied (using unsigned arithmetic):

$$(\text{SUM_SHARED_CREDIT_LIMIT} - \text{SHARED_CUMULATIVE_CREDITS_REQUIRED}) \bmod 2^{\text{[Field Size]}} < (2^{\text{[Field Size]}})/2$$

Equation 2-8 Shared Transmitter Gate non-[Merged] §

- For Posted and Completion credits when [Merged] was requested by the Receiver, Shared Flow Control Usage Limit Enable is Clear and, for each type of information in the TLP, § Equation 2-9 is satisfied (using unsigned arithmetic):

$$\begin{aligned} & (\text{SUM_SHARED_CREDIT_LIMIT_POSTED} \\ & + \text{SUM_SHARED_CREDIT_LIMIT_COMPLETION} \\ & - \text{SHARED_CUMULATIVE_CREDITS_REQUIRED_POSTED} \\ & - \text{SHARED_CUMULATIVE_CREDITS_REQUIRED_COMPLETION}) \bmod 2^{\frac{\text{[Field Size]}}{2}} < (2^{\frac{\text{[Field Size]}}{2}})/2 \end{aligned}$$

Equation 2-9 Shared Transmitter Gate [Merged] §

- For Non-Posted credits and for Posted and Completion credits when [Merged] was not requested by the Receiver, Shared Flow Control Usage Limit Enable is Set and, for each type of information in the TLP, § Equation 2-8 and § Equation 2-10 are both satisfied (using unsigned arithmetic)

$$\begin{aligned} & \text{SCCC} + \text{credit units required for pending TLP} \\ & \leq \text{TOTAL_SHARED_CREDITS_AVAILABLE} \times \text{Shared Flow Control Usage Limit} \times 0.125 \end{aligned}$$

Equation 2-10 Shared Transmitter Usage Limit Gate non-[Merged] §

- For Posted and Completion credits when [Merged] was requested by the Receiver, Shared Flow Control Usage Limit Enable is Set and, for each type of information in the TLP, § Equation 2-9 and § Equation 2-11 are both satisfied (using unsigned arithmetic)

$$\begin{aligned} & \text{SCCC} + \text{credit units required for pending TLP} \\ & \leq (\text{TOTAL_SHARED_CREDITS_AVAILABLE_POSTED} + \\ & \text{TOTAL_SHARED_CREDITS_AVAILABLE_COMPLETION}) \times \text{Shared Flow Control Usage Limit} \times \\ & 0.125 \end{aligned}$$

Equation 2-11 Shared Transmitter Usage Limit Gate [Merged] §

- If the above test does not permit a TLP to be transmitted, continue with the Transmitter gating function test below.
- Shared Flow Control is independent of the VC Arbitration mechanism described in § Section 6.3.3.2 .
- The **Transmitter gating function** test is performed as follows:
 - In Non-Flit Mode, this test applies to all TLPs.
 - In Flit Mode, this test applies to TLPs using dedicated credits. The shared transmitter gating function is used for TLPs using shared credits.
 - For each required type of credit, the number of credits required is calculated as:

$$\text{CUMULATIVE_CREDITS_REQUIRED} = (\text{CREDITS_CONSUMED} + \text{credit units required for pending TLP}) \bmod 2^{\lceil \text{Field Size} \rceil}$$

Equation 2-12 CUMULATIVE_CREDITS_REQUIRED §

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

$$(\text{CREDIT_LIMIT} - \text{CUMULATIVE_CREDITS_REQUIRED}) \bmod 2^{\lceil \text{Field Size} \rceil} \leq 2^{\lceil \text{Field Size} \rceil}/2$$

Equation 2-13 Transmitter Gate §

- If CREDIT_LIMIT was specified as “infinite” during Flow Control initialization, then the gating function is unconditionally satisfied for that type of credit.
- In Flit Mode, the TLP is transmitted with a Flit Mode Local TLP Prefix with the TLP Uses Dedicated Credits bit Set. This indicates that the flit is consuming dedicated credits.
- 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 must not be mixed within one credit.
- When some TLP is blocked from Transmission by a lack of FC Credit, Transmitters must follow the ordering rules specified in § Section 2.4 when determining what types of TLPs must be permitted to bypass the stalled TLP.
- The return of FC credits for a Transaction must not be interpreted to mean that the Transaction has completed or achieved system visibility.
 - Flow Control credit return is used for receive buffer management only, and agents must not make any judgment about the Completion status or system visibility of a Transaction based on the return or lack of return of Flow Control information.
- In Non-Flit Mode, when a Transmitter sends a nullified TLP, the Transmitter does not modify CREDITS_CONSUMED for that TLP (see ¶§ Section 3.6.2.1).
- In Flit Mode, for all TLPs Transmitted, including nullified TLPs, Transmitters must modify CREDITS_CONSUMED ¶§ (when TLP uses dedicated credits) or ¶§ both SHARED_CREDITS_CONSUMED and ¶§ both SHARED_CREDITS_CONSUMED_CURRENTLY ¶§ .. ¶§ (when TLP uses shared credits).

2.6.1.2 FC Information Tracked by Receiver §

- For each type of information tracked, the following quantities are tracked for Flow Control TLP Receiver accounting. In Flit Mode, shared and dedicated credit versions of these are tracked independently.
 - CREDITS_ALLOCATED**
 - Count of the total number of credits granted to the Transmitter since initialization, modulo $2^{\lceil \text{Field Size} \rceil}$ (where [Field Size] is defined in § Table 2-51)
 - Initially set according to the buffer size and allocation policies of the Receiver

Base 6.4 vs Base 6.3

- If [Zero] or [Merged] were advertised by this Receiver, the corresponding CREDITS_ALLOCATED is set to 0
 - This value is included in the InitFC , InitFC1 , InitFC2 , and UpdateFC DLLPs and in the Optimized_Update_FC (see § Section 3.5)
 - Incremented as the Receiver Transaction Layer makes additional receive buffer space available by processing Received TLPs. Optionally permitted to be incremented for dedicated credits or for shared credits when a single VC is using the shared pool, when the Receiver Transaction Layer make additional buffer space available through other mechanisms (e.g., increasing the pool size).
- Updated as shown:

$$\text{CREDITS_ALLOCATED} := (\text{CREDITS_ALLOCATED} + \text{Increment}) \bmod 2^{\lceil \text{Field Size} \rceil}$$

Equation 2-14 CREDITS_ALLOCATED §

(Where *Increment* corresponds to the credits made available, and *[Field Size]* is defined in § Table 2-51)

- For shared credits, CREDITS_ALLOCATED is preserved when VC Enable for VC 1-7 is Cleared and also preserved during subsequent Flow Control initialization.
- **CREDITS RECEIVED**
 - Mandatory for shared credits in Flit Mode.
 - Otherwise, implemented when the optional error check described below is implemented.
 - Count of the total number of FC units consumed by valid TLPs Received since Flow Control initialization, modulo $2^{\lceil \text{Field Size} \rceil}$ (where *[Field Size]* is defined in § Table 2-51)
 - Set to all 0's at interface initialization
 - Updated as shown:

$$\text{CREDITS_RECEIVED} := (\text{CREDITS_RECEIVED} + \text{Increment}) \bmod 2^{\lceil \text{Field Size} \rceil}$$

Equation 2-15 CREDITS_RECEIVED §

(Where *Increment* is the size in FC units of the corresponding part of the received TLP, and *[Field Size]* is defined in § Table 2-51)

for each Received TLP, provided that TLP:

- passes the Data Link Layer integrity checks
- is not malformed or (optionally) is malformed and is not ambiguous with respect to which buffer to release and is mapped to an initialized Virtual Channel
- does not consume more credits than have been allocated (see following rule)

For a TLP with an ECRC Check Failed error, but which otherwise is unambiguous with respect to which buffer to release, CREDITS_RECEIVED MUST@FLIT be updated.

- For shared credits, CREDITS_RECEIVED is preserved when VC Enable for VC 1-7 is Cleared and also preserved during subsequent Flow Control initialization.
- In Flit Mode, the Receiver accounting is modified as follows:

- If the TLP contained a Flit Mode Local TLP Prefix with the TLP Uses Dedicated Credits bit Set, update the dedicated CREDITS_ALLOCATED and dedicated CREDITS_RECEIVED for the associated VC.
- Otherwise, update the shared CREDITS_ALLOCATED and shared CREDITS_RECEIVED for the associated VC.
 - This accounting is not affected by [Merged]. Tracking is independent for Posted and Completion credits.
- Receivers are permitted to optimize their credit return mechanism to return shared and dedicated credits in a different order. For example, if shared TLP S and dedicated TLP D have the same credit type and VC and are received and processed in the order S followed by D, it is permitted to return *dedicated* credits when processing S as long as the corresponding *shared* credits are returned later when processing D.
- In Non-Flit Mode, if a Receiver implements the CREDITS_RECEIVED counter, then when a nullified TLP is received, the Receiver does not modify CREDITS_RECEIVED for that TLP (see [§ 3.6.2.1](#)).
- In Flit Mode, Receivers that implement the CREDITS_RECEIVED counter modify CREDITS_RECEIVED even for nullified TLPS.
- A Receiver may optionally check for Receiver Overflow errors (TLPs exceeding CREDITS_ALLOCATED):
 - For Non-Flit Mode and for dedicated credits in Flit Mode, this is accomplished by checking [§ Equation 2-16](#) using unsigned arithmetic:

$$(CREDITS_ALLOCATED - CREDITS_RECEIVED) \bmod 2^{\lceil \text{Field Size} \rceil} \geq 2^{\lceil \text{Field Size} \rceil}/2$$

Equation 2-16 Receiver Overflow Error Check Non-Flit / Dedicated §

- For shared Non-Posted and for shared Posted and Completion when [Merged] was not advertised by the Receiver, this is accomplished by checking [§ Equation 2-17](#) using unsigned arithmetic:

$$\left| \sum_{i=0}^{i=7} (CREDITS_ALLOCATED[i] - CREDITS_RECEIVED[i]) \right| \bmod 2^{\lceil \text{Field Size} \rceil} \geq \frac{2^{\lceil \text{Field Size} \rceil}}{2}$$

Equation 2-17 Receiver Overflow Error Check Non-Posted / Not [Merged] §

- For shared Posted and Completion when [Merged] was advertised by the Receiver, this is accomplished, by checking [§ Equation 2-18](#) using unsigned arithmetic:

$$\text{temp1}[i] \equiv \text{CREDITS_ALLOCATED}_{\text{POSTED}}[i] + \text{CREDITS_ALLOCATED}_{\text{COMPLETION}}[i]$$

$$\text{temp2}[i] \equiv \text{CREDITS_RECEIVED}_{\text{POSTED}}[i] + \text{CREDITS_RECEIVED}_{\text{COMPLETION}}[i]$$

$$\left| \sum_{i=0}^{i=7} (\text{temp1}[i] - \text{temp2}[i]) \right| \bmod 2^{\text{Field Size}} \geq \frac{2^{\text{Field Size}}}{2}$$

Equation 2-18 Receiver Overflow Error Check [Merged] §

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 that it had allocated for the TLP(s)

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

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

- For non-infinite NPH, NPD, PH, and CplH types, a Flow Control Update must be scheduled for Transmission each time the following events occur: In Non-Flit Mode, a Flow Control Update is an UpdateFC FCP. In Flit Mode, a Flow Control Update is either an UpdateFC FCP (for NPH, NPD, PH, and CplH, both Shared and Dedicated) or an Optimized_Update_FC (for Shared NPH and PH):
 - a. when scaled flow control is not activated and the number of available FC credits of a particular type is zero and one or more units of that type are made available by TLPs processed,
 - b. when scaled flow control is not activated, the NPD credit drops below 2, the Receiver supports either the AtomicOp routing capability or the 128-bit CAS Completer capability, and one or more NPD credits are made available by TLPs processed,
 - c. when scaled flow control is activated and the number of available FC credits of a particular type is zero or is below the scaled threshold and one or more units of that type are made available by TLPs processed so that the number of available credits is equal to or greater than the scaled threshold:
 - For Non-Flit Mode and for dedicated credits in Flit Mode, this threshold is 1 for HdrScale or DataScale of 01b, 4 for HdrScale or DataScale of 10b, and 16 for HdrScale or DataScale of 11b.

- For shared credits in Flit Mode, this threshold is 4 for HdrScale or DataScale of 01b, 4 for HdrScale or DataScale of 10b, and 16 for HdrScale or DataScale of 11b.
- d. when scaled flow control is activated in Non-Flit Mode and for dedicated credits in Flit Mode, the DataScale used for NPD is 01b, the NPD credit drops below 2, the Receiver supports either the AtomicOp routing capability or the 128-bit CAS Completer capability, and one or more NPD credits are made available by TLPs processed.
- e. For shared Non-Posted Data credits in Flit Mode, when the DataScale used for NPD is 01b, the NPD credit drops below 4, and 4 or more NPD credits are made available by TLPs processed.
- For non-infinite PD and CplD types, when the number of available credits is less than the number needed for the Rx_MPS_Limit, a Flow Control Update must be scheduled for Transmission each time one or more units of that type are made available by TLPs processed. In Non-Flit Mode, a Flow Control Update is an UpdateFC FCP. In Flit Mode, a Flow Control Update is either an UpdateFC FCP or an Optimized_Update_FC (for PD type).
 - For a Multi-Function Device where different Functions have different Rx_MPS_Limit values, the largest Rx_MPS_Limit value across all Functions must be used.

When multiple TLPs have been received, and some of the TLP(s) received consumed dedicated credits while other TLP(s) consumed shared credits, the Receiver must return all consumed dedicated credits prior to returning shared credits consumed by TLPs received after the TLP(s) received using dedicated credits. The Receiver is permitted to return the dedicated credits consumed prior to returning shared credits consumed by TLPs received after the TLP(s) using dedicated credits.

IMPLEMENTATION NOTE: RECEIVER HANDLING OF CREDIT RETURN FOR DEDICATED AND SHARED CREDITS §

The purpose of having some amount of Dedicated Credit per VC is to ensure that one VC cannot completely block another VC by consuming all available Shared Credit. To maintain this property it is necessary for the Receiver to ensure that Dedicated Credit is returned in a timely way - such that dedicated credits are returned at or before when the associated TLP(s) is(/are) consumed. In some implementations, buffers may be shared between Dedicated and Shared credits, and the distinction between the two types of Credits lies in how the buffer space is accounted for. In such implementations, it may be desirable to change the accounting for buffer space consumed using shared credits so that dedicated credits can be returned earlier to the Transmitter. To illustrate how this could work, consider the following example - TLPs A, B, C, and D are Received and consumed in that order. C uses dedicated credits while A, B, and D use shared credits.

- If $\text{size}_A \geq \text{size}_C$, the Receiver can return the dedicated credits for C when A is consumed.
- If $\text{size}_B \geq \text{size}_C$, the Receiver can return the dedicated credits for C when B is consumed.
- If $\text{size}_A + \text{size}_B \geq \text{size}_C$, the Receiver can return the dedicated credits for C when B is consumed.
- The Receiver is not permitted to delay the return of the dedicated credits for C to follow the time when D is consumed.

Other rules related to Flow Control:

- UpdateFC FCPs and Optimized_Update_FCs are permitted to be scheduled for Transmission more frequently than is required

- When the Link is in the L0 or L0s Link state, UpdateFC FCPs or Optimized_Update_FC_s for each enabled type of non-infinite FC credit must be scheduled for transmission at least once every 30 µs (-0%/+50%), except in Non-Flit Mode when the Extended Synch bit is Set, in which case the limit is 120 µs (-0%/+50%).
 - This rule is optional when [Zero] dedicated FC credits are required as shown in § Table 3-3 .
 - A timeout mechanism *MUST*@FLIT be implemented. If implemented, such a mechanism must:
 - be active only when the Link is in the L0 or L0s Link state
 - use a timer with a limit of 200 µs (-0%/+50%), where the timer is reset by the receipt of any Init, UpdateFC FCP, or Optimized_Update_FC . Alternately, the timer may be reset by the receipt of any DLLP (see § Section 3.5)
 - upon timer expiration, instruct the Physical Layer to retrain the Link (via the LTSSM Recovery state, § Section 4.2.7.4)
 - in Non-Flit Mode, if an Infinite Credit advertisement has been made during initialization for all three FC types, this timeout mechanism must be disabled for that VC
 - in Flit Mode, if an Infinite Credit advertisement has been made during initialization for all six FC types, this timeout mechanism must be disabled for that VC

IMPLEMENTATION NOTE: USE OF “INFINITE” FC ADVERTISEMENT §

For a given implementation it is possible that not all of the queue types need to be physically implemented in hardware for all Virtual Channels. For example, in a Device that does not support Flit Mode and whose Functions have no AtomicOp Completer, AtomicOp Routing capability, DMWr Completer, or DMWr Routing capability, there is no need to implement a Non-Posted Data queue for Virtual Channels other than VC0, since Non-Posted Requests with data are only allowed on Virtual Channel 0 for such Devices. For unimplemented queues, the Receiver can eliminate the need to present the appearance of tracking Flow Control credits by advertising infinite Flow Control credits during initialization.

IMPLEMENTATION NOTE: NON-FLIT MODE FLOW CONTROL UPDATE LATENCY §

For components subject to receiving streams of TLPs, it is desirable to implement receive buffers larger than the minimum size required to prevent Transmitter throttling due to lack of available credits. Likewise, it is desirable to transmit UpdateFC FCPs such that the time required to send, $\downarrow\downarrow\text{receive}\downarrow\downarrow\uparrow\uparrow\text{receive},\uparrow$ and process the UpdateFC prevents Transmitter throttling. Recommended maximum values for UpdateFC transmission latency during normal operation are shown in § Table 2-52 , § Table 2-53 , and § Table 2-54 . Note that the values given in these tables do not account for any delays caused by the Receiver or Transmitter being in L0s , in Recovery , or for any delays caused by Retimers (see § Section 4.3.9) . For improved performance and/or power-saving, it may be desirable to use a Flow Control update policy that is more sophisticated than a simple timer. Any such policy is implementation specific, and beyond the scope of this document.

The values in the Tables are measured starting from when the Receiver Transaction Layer makes additional receive buffer space available by processing a received TLP, to when the first Symbol of the corresponding UpdateFC DLLP is transmitted.

For a Multi-Function Device where different Functions have different Rx_MPS_Limit values, it is strongly recommended that the smallest Rx_MPS_Limit value across all Functions be used.

*Table 2-52 Maximum UpdateFC Transmission Latency Guidelines
for 2.5 GT/s (Symbol Times) §*

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

*Table 2-53 Maximum UpdateFC Transmission Latency Guidelines
for 5.0 GT/s (Symbol Times) §*

| | | Link Operating Width | | | | | | |
|-------------------------|------|----------------------|------|-----|-----|-----|-----|-----|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Rx_MPS_Limit (bytes) | 128 | 288 | 179 | 124 | 118 | 109 | 99 | 84 |
| | 256 | 467 | 268 | 169 | 158 | 141 | 123 | 96 |
| | 512 | 610 | 340 | 205 | 137 | 160 | 137 | 103 |
| | 1024 | 1122 | 596 | 333 | 201 | 245 | 201 | 135 |
| | 2048 | 2146 | 1108 | 589 | 329 | 416 | 329 | 199 |

| | | Link Operating Width | | | | | | |
|------|------|----------------------|------|-----|-----|-----|-----|-----|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| 4096 | 4194 | 2132 | 1101 | 585 | 757 | 585 | 327 | |

| | | Link Operating Width | | | | | | |
|-------------------------|------|----------------------|------|------|-----|-----|-----|-----|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Rx_MPS_Limit (bytes) | 128 | 333 | 224 | 169 | 163 | 154 | 144 | 129 |
| | 256 | 512 | 313 | 214 | 203 | 186 | 168 | 141 |
| | 512 | 655 | 385 | 250 | 182 | 205 | 182 | 148 |
| | 1024 | 1167 | 641 | 378 | 246 | 290 | 246 | 180 |
| | 2048 | 2191 | 1153 | 634 | 374 | 461 | 374 | 244 |
| | 4096 | 4239 | 2177 | 1146 | 630 | 802 | 630 | 372 |

2.7 End-to-End Data Integrity §

Data integrity across a Link is provided by the Data Link Layer for NFM and by the Physical Layer Logical Block for FM. As TLPs are routed through intermediate components (i.e., Switches) a TLP may become corrupted, and the Link data integrity mechanisms will not detect such corruption. 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 applied to a TLP. The ECRC covers all bits that do not change as the TLP traverses the path (invariant fields). The ECRC is generated by the Transaction Layer in the source component, and checked (if supported) by the ultimate PCI Express Receiver, and optionally by intermediate Receivers. A Switch that supports ECRC checking must check ECRC on TLPs targeting the Switch itself. Such a Switch can optionally check ECRC on TLPs that it forwards. On TLPs that the Switch forwards, the Switch must preserve the error detecting properties of the ECRC, regardless of whether the Switch checks the ECRC, or if the ECRC check fails.⁵⁶

In some cases, the data in a TLP payload is known to be corrupt at the time the TLP is generated, or may become corrupted while passing through an intermediate component, such as a Switch. In these cases, error forwarding, also known as data poisoning, can be used to indicate the corruption to the device consuming the data. In FM, there are two different mechanisms for data poisoning, in support of distinct use models.

2.7.1 ECRC Rules §

The capability to generate and check ECRC is reported to software, and the ability to do so is enabled by software (see § Section 7.8.4.7).

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

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

- For non-IDE TLPs that do not require FM/NFM translation, Switches must pass TLPs with ECRC unchanged from the Ingress Port to the Egress Port⁵⁷
 - For non-IDE TLPs that require FM/NFM translation, Switches must apply ECRC to the translated TLP and must ensure that the error detection capability of ECRC is maintained between the Ingress and Egress Ports; how this is done is outside the scope of this specification.
 - These rules do not apply for IDE TLPs, for which ECRC is not supported and ~~for which~~ FM/NFM translation is not possible.
- If a device supports ECRC generation/checking, at least one of its Functions must support Advanced Error Reporting (AER) (see § [Section 6.2](#))
- If a device Function is enabled to check ECRC, it must do so for all TLPs with ECRC where the device is the ultimate PCI Express Receiver
 - Note that it is still possible for the Function to receive TLPs without ECRC, and these are processed normally - this is not an error

Note that a Switch may optionally perform ECRC checking on TLPs passing through the Switch. ECRC Errors detected by the Switch are reported as described in § [Table 6-5](#), but do not alter the TLPs' passage through the Switch.⁵⁸

A 32-bit ECRC is calculated for the TLP (End-End TLP Prefixes/OHC, header, and data payload), but not, in FM, including any Trailer, using the following algorithm and appended to the end of the TLP (see § [Figure 2-3](#)):

- The ECRC value is calculated using the following algorithm (see § [Figure 2-99](#))
- The polynomial used has coefficients expressed as 04C1 1DB7h
- The seed value (initial value for ECRC storage registers) is FFFF FFFFh
- All header fields, all End-End TLP Prefixes/OHC (if present), and the entire data payload (if present) are included in the ECRC calculation.
- Local TLP Prefixes (if present) are not included in the ECRC calculation.
- All Variant bits must be treated as Set for ECRC calculations.
- In Non-Flit Mode, the following bits are Variant:
 - TLP Header symbol 0, bit 0. This is bit 0 of the Type field⁵⁹. This bit in an End-End TLP Prefix is invariant.
 - TLP Header symbol 2, bit 6. This is either the EP bit or Reserved.
- In Flit Mode, the following bits are Variant:
 - TLP Header symbol 0, bit 0. This is bit 0 of the Type field⁶⁰.
 - TLP Header symbol 6, bit 7. This is either the EP bit or Reserved.
- All other fields are Invariant
- ECRC calculation starts with bit 0 of byte 0 and proceeds from bit 0 to bit 7 of each byte of the TLP
- The result of the ECRC calculation is complemented, and the complemented result bits are mapped into the 32-bit TLP Digest field (NFM), or Trailer (FM), as shown in § [Table 2-55](#).

Table 2-55 Mapping of Bits into ECRC Field §

| ECRC Result Bit | Corresponding Bit Position in the 32-bit TLP ECRC Field |
|-----------------|---|
| 0 | 7 |

57. An exception is a Multicast TLP that an Egress Port is modifying due to the MC_Overlay mechanism. See § [Section 6.14.5](#).

58. An exception is a Multicast TLP that an Egress Port is modifying due to the MC_Overlay mechanism. See § [Section 6.14.5](#).

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

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

Base 6.4 vs Base 6.3

| ECRC Result Bit | Corresponding Bit Position in the 32-bit TLP ECRC Field |
|-----------------|---|
| 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 |

- In NFM, the 32-bit ECRC value is placed in the TLP Digest field at the end of the TLP (see § [Figure 2-3](#)). In FM, the 32-bit ECRC value is placed in the TLP Trailer.
- For TLPs including a TLP Digest field used for an ECRC value, Receivers that support end-to-end data integrity checking check the ECRC value in the TLP Digest field by:
 - applying the same algorithm used for ECRC calculation (above) to the received TLP, not including the 32-bit TLP Digest field of the received TLP, and then:
 - comparing the calculated result with the value in the TLP Digest field of the received TLP.
- Receivers that support end-to-end data integrity checks report violations as an ECRC Error. This reported error is associated with the Receiving Port (see § [Section 6.2](#)).

Beyond the stated error reporting semantics contained elsewhere in this specification, how ultimate PCI Express Receivers make use of the end-to-end data integrity check provided through the ECRC is beyond the scope of this document. Intermediate Receivers are still required to forward TLPs whose ECRC checks fail. A PCI Express-to-PCI/PCI-X Bridge is classified as an ultimate PCI Express Receiver with regard to ECRC checking.

Base 6.4 vs Base 6.3

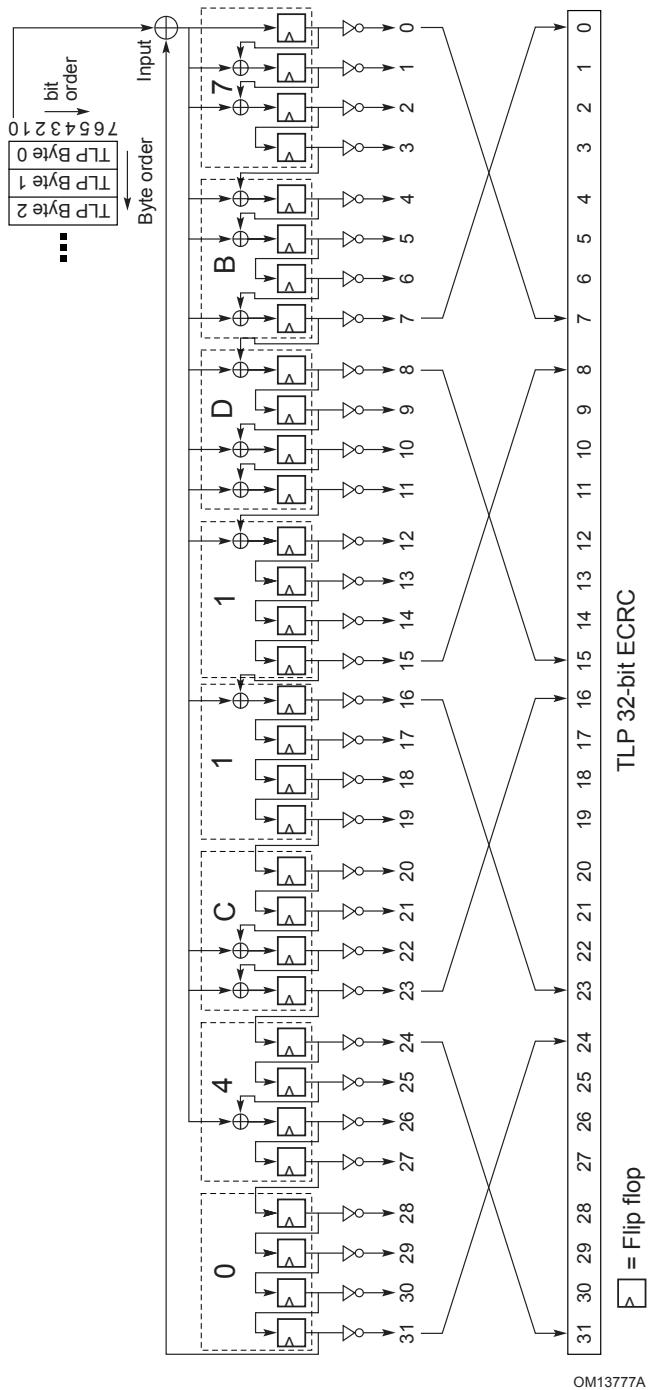


Figure 2-99 Calculation of 32-bit ECRC for TLP End to End Data Integrity Protection §

IMPLEMENTATION NOTE: PROTECTION OF TD BIT INSIDE SWITCHES (NFM) §

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

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

IMPLEMENTATION NOTE: DATA LINK LAYER DOES NOT HAVE INTERNAL TLP VISIBILITY (NFM) §

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

2.7.2 Error Forwarding (Data Poisoning) §

Error Forwarding (also known as data poisoning), is indicated by Setting the EP bit, or additionally, in FM, through the use of Physical Layer Logical Block mechanisms. In FM, either or both mechanisms are permitted to be applied to a TLP with a data payload, and the requirements defined in this specification for Receiver handling of poisoned TLPs are the same regardless of the poisoning mechanism applied. It is permitted for Receivers to additionally implement differentiated handling based on the type of poisoning mechanism applied, but such handling is outside the scope of this specification.

The rules for the use of the EP bit are specified in § Section 2.7.2.1 . The rules for the use of Physical Layer Logical Block mechanisms for data poisoning are specified in § Section 4.2.3.4 . Here are some examples of cases where Error Forwarding might be used:

- Example #1: A read from parity or ECC-protected memory encounters an uncorrectable error (EP bit)
- Example #2: An error detected at the source of a write directed towards system memory (EP bit)
- Example #3: Data integrity error on an internal data buffer or cache within a routing element (EP bit or Physical Layer Logical Block)

Considerations for the use of Error Forwarding

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

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

2.7.2.1 Rules For Use of Data Poisoning §

- Support for TLP poisoning in a Transmitter is optional.
 - In FM, a Transmitter is permitted to support only the EP bit mechanism, only the Physical Layer Logical Block mechanism, or both, or neither.
- Data poisoning applies only to the data payload⁶¹ within a Write Request (Posted or Non-Posted), a Message with Data, an AtomicOp Request, a Read Completion, or an AtomicOp Completion.
- Poisoning of a TLP with a data payload in the Transaction Layer is indicated by a Set EP bit.
- When a routing element is translating a TLP from NFM to FM, if the EP bit is Set in the NFM TLP, then the EP bit must be Set in the FM TLP.
- When a routing element is translating a TLP from FM to NFM, if either poisoning mechanism has been applied to the FM TLP, then the EP bit must be Set in the NFM TLP.
- Transmitters are only permitted to poison TLPs that include a data payload. In FM, the EP bit is Reserved for TLPs that do not include a data payload. In NFM, the behavior of the Receiver is not specified if poisoning is indicated for any TLP that does not include a data payload.
- For IDE TLPs:
 - Only the original Transmitting Port is permitted to poison a TLP and must do so using the EP ~~↑↓bit↓~~
↑↓bit mechanism.↑
 - It is not permitted to use Physical Layer Logical Block mechanisms to poison a TLP; if data corruption is detected in an IDE TLP after the time the MAC has been generated, the IDE TLP must be forwarded without consideration of the detected corruption.
- If a Transmitter supports data poisoning, TLPs that are known at the Transaction Layer of the Transmitter to include a bad data payload must use the EP bit ~~↑↓poison↓~~ mechanism.
- For a routing element that supports data poisoning, if a non-IDE TLP is Received as poisoned using Physical Layer Logical Block mechanisms, that TLP must be transmitted at the Egress Port marked as poisoned using the EP bit mechanism.
- If a Downstream Port supports Poisoned TLP Egress Blocking, the Poisoned TLP Egress Blocking Enable bit is Set, and a poisoned TLP targets going out the Egress Port, the Port must handle the TLP as a Poisoned TLP Egress Blocked error unless there is a higher precedence error. See § Section 6.2.3.2.3, § Section 6.2.5, and § Section 7.9.14.3. Further:
 - The Port must not transmit the TLP.
 - If DPC is not triggered and the TLP is a Non-Posted Request received on a non-UIO VC, the Port must return a Completion with Unsupported Request Completion Status. See § Section 6.2.3.2.4.1.
 - If DPC is triggered the Port must behave as described in § Section 2.9.3.
- For ultimate Completers:
 - The following Requests with poisoned data payload must not modify the value of the target location:
 - Configuration Write Request

61. A data payload includes TLPs with no bytes enabled (e.g., zero-length writes).

- Any of the following that target a control register or control structure in the Completer: I/O Write Request, Memory Write Request, or ~~↓↑non-vendor defined↓~~
~~↑↓non-Vendor-Defined↑~~ Message with data
- AtomicOp Request
- DMWr Request (see § Section 6.32 ~~↓↑)↑~~)

Unless there is a higher precedence error, a Completer must handle these Requests as a Poisoned TLP Received error⁶², and the Completer must also return a Completion with a Completion Status of Unsupported Request (UR) if the Request is Non-Posted ~~↓↑or UIO↑~~ (see § Section 6.2.3.2.3, § Section 6.2.3.2.4, and § Section 6.2.5). Regardless of the severity of the reported error, the reported error must be handled as an uncorrectable error, not an Advisory Non-Fatal Error.

Errata: Base 6.3
B834Δ~~↓~~

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

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

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

2.8 Completion Timeout Mechanism §

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

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

The Completion Timeout mechanism may be disabled by configuration software by means of the Completion Timeout Disable mechanism (see § Section 7.5.3.15 and § Section 7.5.3.16).

The Completion Timeout limit is set in the Completion Timeout Value field of the Device Control 2 register. A Completion Timeout is a reported error associated with the Requester Function (see § Section 6.2). If the Completion Timeout programming mechanism is not supported, the Function MUST@FLIT implement a timeout value in the range 40 ms to 50 ms; when Flit Mode Supported is Clear, the Function must implement a timeout value in the range 50 µs to 50 ms, and it is strongly recommended that the value be at least 10 ms.

A Request for which there are multiple Completions must be considered completed only when all Completions have been received by the Requester.

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

For a Memory Read Request, if some, but not all, requested data is returned before the Completion Timeout timer expires, the Requester is permitted to keep or to discard the data that was returned prior to timer expiration.

Completion Timeout expiration for a UIO Request does not necessarily indicate that the Request, or portions of the Request, succeeded or failed.

For a series of UIO Requests using the same Transaction ID, the Completion Timeout mechanism must be restarted for each UIO Request issued.

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

IMPLEMENTATION NOTE: COMPLETION TIMEOUT PREFIX/HEADER LOG CAPABLE §

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

A Completion Timeout may be the result of improper configuration, system failure, or async removal (see § Section 6.7.6). In order for host software to distinguish a Completion Timeout error after which continued normal operation is not possible (e.g., after one caused by improper configuration or a system failure) from one where continued normal operation is possible (e.g., after an async removal), it is strongly encouraged that Requesters log the Request TLP prefix/header associated with the Completion Timeout.

2.9 Link Status Dependencies §

2.9.1 Transaction Layer Behavior in DL_Down Status §

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

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

For a Downstream Port, DL_Down status is handled by:

- Initializing back to their default state any buffers or internal states associated with outstanding requests transmitted Downstream

- Port configuration registers must not be affected, except as required to update status associated with the transition to DL_Down.
- For Non-Posted ↑↑Requests and UIO↑ Requests, forming completions for any Requests submitted by the device core for Transmission, returning Unsupported Request Completion Status, then discarding the Requests
 - This is a reported error associated with the Function for the (virtual) Bridge associated with the Port (see § Section 6.2). For Root Ports, the reporting of this error is optional.
 - Non-Posted Requests ↑↑and UIO Requests↑ already being processed by the Transaction Layer, for which it may not be practical to return Completions, are discarded.

 Errata: Base 6.3
 B834△◀▶

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

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

Note: The point at which a Non-Posted Request ↑↑or UIO Request↑ becomes “uncompletable” is implementation specific.

- The Port must terminate any PME_Turn_Off handshake Requests targeting the Port in such a way that the Port is considered to have acknowledged the PME_Turn_Off request (see the Implementation Note in § Section 5.3.3.2.1).
- The Port must handle Vendor-Defined Message Requests as described in § Section 2.2.8.6 (e.g., silently discard Vendor-Defined Type 1 Messages Requests that it is not designed to receive) since the DL_Down prevents the Request from reaching its targeted Function.
- For all other Posted Requests, discarding the Requests
 - This is a reported error associated with the Function for the (virtual) Bridge associated with the Port (see § Section 6.2), and must be reported as an Unsupported Request. For Root Ports, the reporting of this error is optional.
 - For a Posted Request already being processed by the Transaction Layer, the Port is permitted not to report the error.

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

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

- Discarding all Completions submitted by the device core for Transmission

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

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

2.9.2 Transaction Layer Behavior in DL_Up Status §

DL_Up status indicates that a connection has been established with another component on the associated Link. This section specifies the Transaction Layer's behavior when the Data Link Layer reports entry to the DL_Up status to the

Transaction Layer, indicating that the Link is operational. The Transaction Layer of a Port with DL_Up status must accept received TLPs that conform to the other rules of this specification.

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

- When transitioning from a non- DL_Up status to a DL_Up status and the Auto Slot Power Limit Disable bit is Clear in the Slot Control Register, the Port must initiate the transmission of a Set_Slot_Power_Limit Message to the other component on the Link to convey the value programmed in the Slot Power Limit Scale and Slot Power Limit Value fields of the Slot Capabilities Register. This Transmission is optional if the Slot Capabilities Register has not yet been initialized.

2.9.3 Transaction Layer Behavior During Downstream Port Containment §

During Downstream Port Containment (DPC), the LTSSM associated with the Downstream Port is directed to the Disabled state. Once it reaches the Disabled state, it remains there as long as the DPC Trigger Status bit in the DPC Status Register is Set. See § Section 6.2.11 for requirements on how long software must leave the Downstream Port in DPC. This section specifies the Transaction Layer's behavior once DPC has been triggered, and as long as the Downstream Port remains in DPC.

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

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

- If the condition that triggered DPC was associated with a Downstream TLP, any prior Downstream TLPs are permitted to be dropped silently or transmitted before the Link goes down. Otherwise, the following rules apply.
 - For each Non-Posted ↑↑Request or UIO↑ Request, the Port must return a Completion and discard the Request silently. The Completer ID field must contain the value associated with the Downstream Port.
 - If the DPC Completion Control bit is Set in the DPC Control Register, then Completions are generated with Unsupported Request (UR) Completion Status.
 - If the DPC Completion Control bit is Clear, Completions are generated with Completer Abort (CA) Completion Status.
 - The Port must terminate any PME_Turn_Off handshake Requests targeting the Port in such a way that the Port is considered to have acknowledged the PME_Turn_Off Request (see the Implementation Note in § Section 5.3.3.2.1).
 - The Port must handle Vendor-Defined Message Requests as described in § Section 2.2.8.6. (e.g., silently discard Vendor Defined_Type 1 Message Requests that it is not designed to receive) since the DL_Down prevents the Request from reaching its targeted Function.
 - For all other Posted Requests and Completions, the Port must silently discard the TLP.

Errata: Base 6.3
B834△↔

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

Errata: Base 6.3
B834△↔

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

Errata: Base 6.3
B834△↔

[UIO Requests](#) that get tracked is implementation specific, but it is strongly recommended that all Non-Posted Requests [\[1\]](#) and [UIO Requests](#) that are generated by host processor instructions (e.g., “read”, “write”, “load”, “store”, or one that corresponds to an AtomicOp) be tracked. Other candidates for tracking include peer-to-peer Requests coming from other Root Ports and Requests coming from RCiEPs.

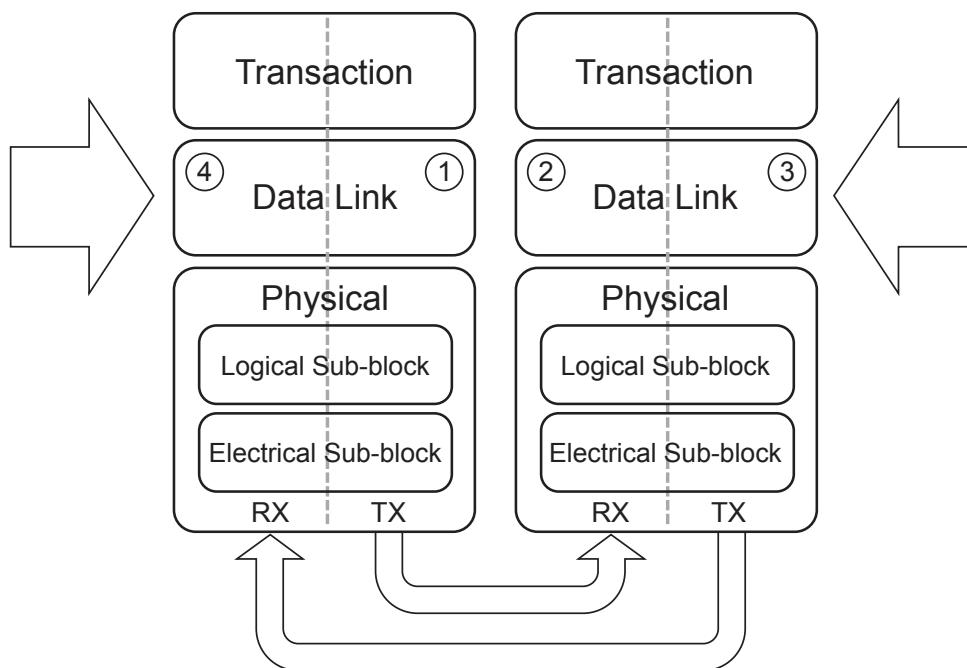
- Otherwise, the associated Requesters may encounter Completion Timeouts. The software solution stack should comprehend and account for this possibility.

Base 6.4 vs Base 6.3

3. Data Link Layer Specification §

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

3.1 Data Link Layer Overview §



OM13778A

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

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

Data Exchange:

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

Error Detection and Retry (Non-Flit Mode):

- TLP Sequence Number and LCRC generation
- Transmitted TLP storage for Data Link Layer Retry

- Data integrity checking for TLPs and Data Link Layer Packets (DLLPs)
- Positive and negative acknowledgement DLLPs
- Error indications for error reporting and logging mechanisms
- Link Acknowledgement Timeout replay mechanism

Initialization and power management:

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

DLLPs are:

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

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

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

- Received DLLPs that fail the CRC check are discarded. The mechanisms that use DLLPs may suffer a performance penalty from this loss of information, but are self-repairing such that a successive DLLP will supersede any information lost.
- TLPs that fail the data integrity checks (LCRC and \downarrow sequence number), \downarrow \uparrow TLP Sequence Number, \uparrow or that 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.

In Flit Mode, both DLLPs and TLPs are sent using Flits. Flits contain the data integrity checks \downarrow LCRC, \downarrow \uparrow CRC, \uparrow FEC, and \downarrow sequence number, \downarrow \uparrow Flit Sequence Number, \uparrow . Replay occurs at the Flit level (see § [Section 4.2.3.4](#) and § [Section 4.2.3.4.2.1](#)).

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

3.2 Data Link Control and Management State Machine §

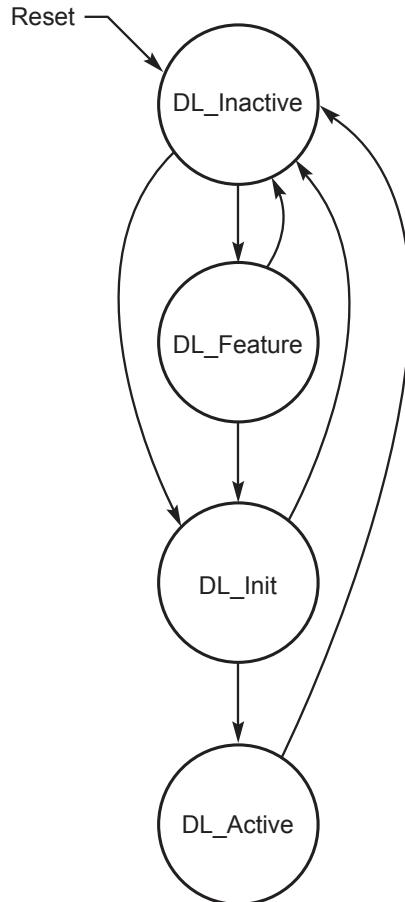
The Data Link Layer tracks the state of the Link. It communicates Link status with the Transaction and Physical Layers, and performs Link management through the Physical Layer. The Data Link Layer contains the Data Link Control and Management State Machine (DLCMSM) to perform these tasks. The states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

States:

- Base 6.4 vs Base 6.3
- DL_Inactive - Physical Layer reporting Link is non-operational or nothing is connected to the Port
 - DL_Feature (optional) - Physical Layer reporting Link is operational, perform the Data Link Feature Exchange
 - DL_Init - Physical Layer reporting Link is operational, initialize Flow Control for the default Virtual Channel
 - DL_Active - Normal operation mode

Status Outputs:

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



OM13779A

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

3.2.1 Data Link Control and Management State Machine Rules §

Rules per state:

- ***DL_Inactive***
 - Initial state following PCI Express Hot , Warm , or Cold Reset (see § Section 6.6). Note that DL states are unaffected by an FLR (see § Section 6.6).

- Upon entry to DL_Inactive
 - Reset all Data Link Layer state information to default values
 - If the Port supports the optional Data Link Feature Exchange, the Remote Data Link Feature Supported , and Remote Data Link Feature Supported Valid fields must be $\downarrow\downarrow$ cleared. $\downarrow\uparrow$
 $\uparrow\downarrow$ Cleared. $\downarrow\uparrow$
 - Discard the contents of the Data Link Layer Retry Buffer (see § Section 3.6)
- While in DL_Inactive :
 - Report DL_Down status to the Transaction Layer as well as to the rest of the Data Link Layer

Note: This will cause the Transaction Layer to discard any outstanding transactions and to terminate internally any attempts to transmit a TLP. For a Downstream Port, this is equivalent to a Hot-Remove. For an Upstream Port, having the Link go down is equivalent to a $\downarrow\downarrow$ hot reset $\downarrow\uparrow$
 $\uparrow\downarrow$ Hot Reset $\downarrow\uparrow$ (see § Section 2.9).
 - Discard TLP information from the Transaction and Physical Layers
 - Do not generate or accept DLLPs
- Exit to DL_Feature if all of the following conditions are satisfied :
 - the Port supports Data Link Feature Exchange
 - either the Data Link Feature Exchange is Enabled bit in the Data Link Feature Extended Capability is Set or the Data Link Feature Extended Capability is not implemented
 - the Transaction Layer indicates that the Link is not disabled by software
 - the Physical Layer reports Physical LinkUp = 1b
- Exit to DL_Init if:
 - The Port does not support the optional Data Link Feature Exchange, the Transaction Layer indicates that the Link is not disabled by software, and the Physical Layer reports Physical LinkUp = 1b
or
 - The Port supports the optional Data Link Feature Exchange, the Data Link Feature Exchange is Enabled bit is Clear, the Transaction Layer indicates that the Link is not disabled by software, and the Physical Layer reports Physical LinkUp = 1b
- **DL_Feature**
 - While in DL_Feature:
 - Perform the Data Link Feature Exchange protocol as described in § Section 3.3
 - Report DL_Down status
 - The Data Link Layer of a Port with DL_Down status is permitted to discard any received TLPs provided that it does not acknowledge those TLPs by sending one or more Ack DLLPs
 - Exit to DL_Init if:
 - Data Link Feature Exchange completes successfully, and the Physical Layer continues to report Physical LinkUp = 1b,
or
 - Data Link Feature Exchange determines that the remote Data Link Layer does not support the optional Data Link Feature Exchange protocol, and the Physical Layer continues to report Physical LinkUp = 1b
 - Terminate the Data Link Feature Exchange protocol and exit to DL_Inactive if:
 - Physical Layer reports Physical LinkUp = 0b

• ***DL_Init***

- While in DL_Init :
 - Initialize Flow Control for the default Virtual Channel, VC0, following the Flow Control initialization protocol described in § Section 3.4
 - Report DL_Down status while in state FC_INIT1 ; DL_Up status while in state FC_INIT2
 - The Data Link Layer of a Port with DL_Down status is permitted to discard any received TLPs provided that it does not acknowledge those TLPs by sending one or more Ack DLLPs
- Exit to DL_Active if:
 - Flow Control initialization completes successfully, and the Physical Layer continues to report Physical LinkUp = 1b
- Terminate attempt to initialize Flow Control for VC0 and exit to DL_Inactive if:
 - Physical Layer reports Physical LinkUp = 0b

• ***DL_Active***

- DL_Active is referred to as the normal operating state
- While in DL_Active :
 - Accept and transfer TLP information with the Transaction and Physical Layers as specified in this chapter
 - Generate and accept DLLPs as specified in this chapter
 - Report DL_Up status to the Transaction and Data Link Layers
- Exit to DL_Inactive if:
 - Physical Layer reports Physical LinkUp = 0b
 - Downstream Ports that are Surprise Down Error Reporting Capable (see § Section 7.5.3.6) must treat this transition from DL_Active to DL_Inactive as a Surprise Down error, except in the following cases where this error detection is blocked:
 - If the Secondary Bus Reset bit in the Bridge Control Register has been Set by software, then the subsequent transition to DL_Inactive must not be considered an error.
 - If the Link Disable bit has been Set by software or if DPC has been triggered, then the subsequent transition to DL_Inactive must not be considered an error.
 - If a Switch Downstream Port transitions to DL_Inactive due to an event above that Port, that transition to DL_Inactive must not be considered an error. Example events include the Switch Upstream Port propagating Hot Reset, the Switch Upstream Link transitioning to DL_Down, and the Secondary Bus Reset bit in the Switch Upstream Port being Set.
 - If a PME_Turn_Off Message has been sent through this Port, then the subsequent transition to DL_Inactive must not be considered an error.
 - Note that the DL_Inactive transition for this condition will not occur until a power off, a reset, or a request to restore the Link is sent to the Physical Layer.
 - Note also that in the case where the PME_Turn_Off / PME_TO_Ack handshake fails to complete successfully, a Surprise Down error may be detected.
 - If the Port is associated with a hot-pluggable slot (the Hot-Plug Capable bit in the Slot Capabilities register Set), and the Hot-Plug Surprise bit in the Slot Capabilities register is Set, then any transition to DL_Inactive must not be considered an error.

- If the Port is associated with a hot-pluggable slot (Hot-Plug Capable bit in the Slot Capabilities register Set), and Power Controller Control bit in Slot Control register is Set (Power-Off), then any transition to DL_Inactive must not be considered an error.

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

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

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

IMPLEMENTATION NOTE: PHYSICAL LAYER THROTTLING §

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

3.3 Data Link Feature Exchange §

The Data Link Feature Exchange protocol is required for Ports that support Flit Mode and for Ports that support 16.0 GT/s and higher data rates. It is optional for other Ports. Downstream Ports that implement this protocol must contain the Data Link Feature Extended Capability (see § Section 7.7.4). Upstream Ports that implement this protocol are optionally permitted to include the Data Link Feature Extended Capability. This capability contains four fields:

- The Local Data Link Feature Supported field indicates the Data Link Features supported by the local Port
- The Remote Data Link Feature Supported field indicates the Data Link Features supported by the remote Port
- The Remote Data Link Feature Supported Valid bit indicates that the Remote Data Link Feature Supported field contains valid data
- The Data Link Feature Exchange is Enabled field permits systems to disable the Data Link Feature Exchange. This can be used to work around legacy hardware that does not correctly ignore the DLLP.

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

Rules for this protocol are:

- On entry to DL_Feature:
 - It is permitted to Clear the Remote Data Link Feature Supported and Remote Data Link Feature Supported Valid fields
- While in DL_Feature:
 - Transaction Layer must block transmission of TLPs
 - Transmit the Data Link Feature DLLP

- The transmitted Feature Supported field must use the value in the Local Data Link Feature Supported field.
- The transmitted Feature Ack bit must use the value in the Remote Data Link Feature Supported Valid bit.
- The Data Link Feature DLLP must be transmitted at least once every 34 µs. Time spent in the Recovery or Configuration LTSSM states does not contribute to this limit.
 - Process received Data Link Feature DLLPs :
 - If the Remote Data Link Feature Supported Valid bit is Clear, record the Feature Supported field from the received Data Link Feature DLLP in the Remote Data Link Feature Supported field and Set the Remote Data Link Feature Supported Valid bit.
- Exit DL_Feature if:
 - An InitFC1 DLLP has been received.
 - An MR-IOV MRIInit DLLP (encoding 0000 0001b) has been received. MR-IOV is deprecated so this clause has no effect in new designs.
or
 - While in DL_Feature, at least one Data Link Feature DLLP has been received with the Feature Ack bit Set.

A **Data Link Feature** is a field representing a protocol feature. Protocol features are either activated, not activated, or not supported. A Data Link Feature is activated when Remote Data Link Feature Supported Valid is Set and when the associated Feature Supported bit is Set in both the Local Data Link Feature Supported and Remote Data Link Feature Supported fields.

A **Data Link Parameter** is a field that communicates a value across the interface. Data Link Parameters have a parameter-specific mechanism for software to determine when the field value is meaningful. For example, the field might be meaningful if its value is non-zero or the field might be meaningful if some other field(s) have specific values.

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

Table 3-1 Data Link Feature Supported Bit Definition §

| <u>Bit Location</u> | Description | Type |
|---------------------|---|----------------------------|
| 0 | <p>Scaled Flow Control – indicates support for Scaled Flow Control.</p> <p>Must be Set in Ports that support 16.0 GT/s or higher data rates.</p> <p>Must be Set if Flit Mode is enabled.</p> | <u>Data Link Feature</u> |
| 1 | <p>Immediate Readiness – indicates that all non-Virtual Functions in the sending Port have <u>Immediate Readiness Set</u> (see § Section 7.5.1.1.4).</p> <p>In Flit Mode, this bit is always meaningful. In Non-Flit Mode, this bit is meaningful when Set, but when Clear indicates either that some non-Virtual Function has <u>Immediate Readiness Clear</u> or that the sending Port is not providing this information.</p> | <u>Data Link Parameter</u> |
| 4:2 | <p>Extended VC Count – This field indicates the number of VC Resources supported by the sending Port. This is the value of the <u>Extended VC Count</u> field in either the <u>Multi-Function Virtual Channel Extended Capability</u> or the <u>Virtual Channel Extended Capability</u> (with Capability ID 0002h).</p> <p>This field is meaningful in Flit Mode. In Non-Flit Mode, this field must be zero.</p> | <u>Data Link Parameter</u> |

| ↓↑Bit Location↓ ↑↓Bit Location↑ | Description | Type | | | | | | | | | | | | | | | | |
|--|---|-------------|----------------|-------------|------------------------|-------------|------------------------|-------------|------------------------|-------------|-------------------------|-------------|--------------------------|-------------|-------------|-------------|-----------------|---------------------|
| 7:5 | <p>L0p Exit Latency - This field indicates ↑↑the↑ sending Port's L0p Exit Latency. The value reported indicates the length of time the sending Port requires to complete widening a link using L0p. If the sending Port does not support L0p, this field must contain 000b.</p> <p>Defined encodings are:</p> <table> <tr><td>000b</td><td>Less than 1 µs</td></tr> <tr><td>001b</td><td>1 µs to less than 2 µs</td></tr> <tr><td>010b</td><td>2 µs to less than 4 µs</td></tr> <tr><td>011b</td><td>4 µs to less than 8 µs</td></tr> <tr><td>100b</td><td>8 µs to less than 16 µs</td></tr> <tr><td>101b</td><td>16 µs to less than 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> <p>This field is meaningful in Flit Mode. In Non-Flit Mode, this field must be ↓↓zero.↓↑Zero.↑</p> | 000b | Less than 1 µs | 001b | 1 µs to less than 2 µs | 010b | 2 µs to less than 4 µs | 011b | 4 µs to less than 8 µs | 100b | 8 µs to less than 16 µs | 101b | 16 µs to less than 32 µs | 110b | 32 µs-64 µs | 111b | More than 64 µs | Data Link Parameter |
| 000b | Less than 1 µs | | | | | | | | | | | | | | | | | |
| 001b | 1 µs to less than 2 µs | | | | | | | | | | | | | | | | | |
| 010b | 2 µs to less than 4 µs | | | | | | | | | | | | | | | | | |
| 011b | 4 µs to less than 8 µs | | | | | | | | | | | | | | | | | |
| 100b | 8 µs to less than 16 µs | | | | | | | | | | | | | | | | | |
| 101b | 16 µs to less than 32 µs | | | | | | | | | | | | | | | | | |
| 110b | 32 µs-64 µs | | | | | | | | | | | | | | | | | |
| 111b | More than 64 µs | | | | | | | | | | | | | | | | | |
| 22:8 | Reserved | | | | | | | | | | | | | | | | | |

3.4 Flow Control Initialization Protocol §

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

Shared Flow Control is enabled in Flit Mode. Shared Flow Control is disabled in Non-Flit Mode.

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

- FC_INIT1
- FC_INIT2

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

3.4.1 Flow Control Initialization State Machine Rules §

- If at any time during initialization for VCs 1-7 the VC is disabled, the flow control initialization process for the VC is terminated
- Rules for state **FC_INIT1** :
 - Entered when initialization of a VC (VCx) is required
 - When the DL_Init state is entered ($VCx = VC0$)
 - When VC ($VCx = VC1-7$) is enabled by software (see § Section 7.9.1 and § Section 7.9.2)

- While in FC_INIT1:
 - Transaction Layer must block transmission of TLPs using VCx
 - In Non-Flit Mode, transmit the following three InitFC1 DLLPs for VCx in the following relative order:
 - InitFC1-P [Dedicated] (first)
 - InitFC1-NP [Dedicated] (second)
 - InitFC1-Cpl [Dedicated] (third)
 - In Flit Mode, transmit the following six InitFC1 DLLPs for VCx in the following relative order:
 - InitFC1-P [Dedicated] (first)
 - InitFC1-NP [Dedicated] (second)
 - InitFC1-Cpl [Dedicated] (third)
 - InitFC1-P [Shared] (fourth)
 - InitFC1-NP [Shared] (fifth)
 - InitFC1-Cpl [Shared] (sixth)
 - The three (or six) InitFC1 DLLPs must be transmitted at least once every 34 µs.
 - Time spent in the Recovery or Configuration LTSSM states does not contribute to this limit.
 - It is strongly ~~↓↑encouraged↓~~ ↑↑recommended↑ that the InitFC1 DLLP transmissions are repeated frequently, particularly when there are no other TLPs or DLLPs available for transmission.
 - Set DataFC, DataScale, HdrFC, and HdrScale as shown in § Table 3-2 and § Table 3-3
 - When DataFC or HdrFC is 0, the following encodings of DataScale and HdrScale are used:
 - Non-Flit Mode, no Scaled FC:
[Infinite.1]
 $\text{DataScale/HdrScale} = 00\text{b}$

[Reserved]
 $\text{DataScale/HdrScale} = 01\text{b}, 10\text{b}, \text{ or } 11\text{b}$

- Non-Flit Mode, Scaled FC:
[Infinite.2]
 $\text{DataScale/HdrScale} = 01\text{b}, 10\text{b}, \text{ or } 11\text{b}$

[Reserved]
 $\text{DataScale/HdrScale} = 00\text{b}$

- Flit Mode:
[Infinite.3]
 $\text{DataScale/HdrScale} = 00\text{b}$

[Zero]
 $\text{DataScale/HdrScale} = 01\text{b}$

[Merged]
 $\text{DataScale/HdrScale} = 10\text{b}$

[Reserved]
 $\text{DataScale/HdrScale} = 11\text{b}$

Base 6.4 vs Base 6.3

Table 3-2 InitFC1 / InitFC2 Options – Non-Flit Mode §

| Row | Scaled Flow Control Supported and Activated | Local and Remote ↑↓Extended VC Count↓ ↑↓Extended VC Count↑ | Merged | InitFC Count | Dedicated / Shared | DataScale and HdrScale | DataFC and HdrFC | Notes |
|-----|---|--|----------------|--------------|--------------------|------------------------|------------------|-------|
| 1 | No Scaled FC | 0 (Reserved) | Not Applicable | 3 | 0b (Reserved) | [Infinite.1] | | 1 |
| | | | | | | 00b | ≠ 0 | |
| 2 | Scaled FC | 0 (Reserved) | Not Applicable | 3 | 0b (Reserved) | [Infinite.2] | | 1, 2 |
| | | | | | | {01b 10b 11b} | ≠ 0 | |

Notes:

1. Backwards compatibility: In earlier versions of this specification, the DataScale and HdrScale bits are Reserved.
2. Backwards compatibility: In earlier versions of this specification, any non-zero DataScale/HdrScale means [Infinite.2].

Table 3-3 InitFC1 / InitFC2 Options – Flit Mode §

| Row | Local and Remote ↑↓Extended VC Count↓ ↑↓Extended VC Count↑ | Merged | InitFC Count | Dedicated / Shared | Kind | DataScale and HdrScale | DataFC and HdrFC | Notes |
|-----|--|------------|--------------|--------------------|------------------|------------------------|------------------|---------|
| 3 | Local = 0 OR optionally (Local ≠ 0 and Remote = 0) | Not Merged | 6 | 0b (shared) | P, NP, Cpl | [Infinite.3] | | 5, 9 |
| | | | | | | {01b 10b 11b} | ≠ 0 | |
| 4 | Remote = 0) | Merged | 6 | 1b (dedicated) | P, NP, Cpl | [Zero] | | 3, 5, 9 |
| | | | | | | | | |
| 5 | Local = 0 OR optionally (Local ≠ 0 and Remote = 0) | Merged | 6 | 0b (shared) | P, NP | [Infinite.3] | | 8 |
| | | | | | | {01b 10b 11b} | ≠ 0 | |
| 6 | Local = 0 OR optionally (Local ≠ 0 and Remote = 0) | Not Merged | 6 | Cpl | | [Merged] | | 7, 8 |
| | | | | | | | | |
| 7 | Local = 0 OR optionally (Local ≠ 0 and Remote = 0) | Merged | 6 | 1b (dedicated) | P, Cpl | [Infinite.3] | | 8, 10 |
| | | | | | | {01b 10b 11b} | ≠ 0 | |
| 8 | Local ≠ 0 | Not Merged | 6 | NP | | [Zero] | | 4, 8 |
| | | | | | | | | |
| 9 | Local ≠ 0 | Not Merged | 6 | 0b (shared) | P, NP, Cpl | [Infinite.3] | | 5, 6 |
| | | | | | | [Zero] | | |
| | | | | | | {01b 10b 11b} | ≠ 0 | |

Base 6.4 vs Base 6.3

| Row | Local and Remote ↑↓Extended VC Count↓ ↑↑Extended VC Count↑ | Merged | InitFC Count | Dedicated / Shared | Kind | DataScale and HdrScale | DataFC and HdrFC | Notes |
|-----|--|--------|--------------|--------------------|------------------|---|------------------|-------|
| 10 | | | | 1b (dedicated) | P, NP, Cpl | [Infinite.3] { 01b 10b 11b } ≠ 0 | | 5 |
| 11 | | | | 0b (shared) | P, NP | [Infinite.3] [Zero] { 01b 10b 11b } ≠ 0 | | 6, 8 |
| 12 | Local ≠ 0 | Merged | 6 | | Cpl | [Merged] | | 7, 8 |
| 13 | | | | 1b (dedicated) | P, NP, Cpl | [Infinite.3] { 01b 10b 11b } ≠ 0 | | 8, 10 |

Notes:

3. Since Extended VC Count is 0, only 1 VC is possible. [Zero] dedicated credits allocated. All credits are Shared.
4. Since Extended VC Count is 0, only 1 VC is possible. [Zero] Non-Posted dedicated credits are allocated and all Non-Posted credits are Shared.
5. When (Local ≠ 0 and Remote = 0), rows 3-4 are recommended but rows 9-10 are permitted.
6. [Zero] shared credits indicates that no additional shared credits are being allocated. All shared credits are (were) allocated by other VCs (and are usable by this VC).

When more than one VC advertises shared credits with scale factor 01b, 10b, or 11b, that scale factor must be able to express all allocated shared credits, regardless of VC. For example, if VC0 and VC1 each advertise 120 header credits (i.e., a total of 240 credits), they must do so using a scale factor other than 1 (01b) since that scale factor is limited to 127 outstanding credits.

Use of [Infinite.3] shared credits must be consistent across VCs. If one VC uses [Infinite.3] shared credits for a given credit type (P/NP/Cpl, Hdr/Data), all VCs must also use [Infinite.3] shared credits for that credit type.

Use of scale values (HdrScale or DataScale) 01b, 10b, or 11b for shared credits must be consistent across VCs. If one VC uses scale value 01b, 10b, or 11b for a given credit type (P/NP/Cpl, Hdr/Data), all other VCs must either use the same scale value or must use [Zero] for that credit type.

When one VC uses [Zero] shared credits for a given credit type (P/NP/Cpl, Hdr/Data), at least one VC must use a scale value of 01b, 10b, or 11b for that credit type. For that credit type, shared credit UpdateFC DLLPs for all VCs must use this non-00b scale value, including VCs that advertised [Zero] during initialization.

It is permitted for all VCs to offer [Zero] shared credits resulting in only dedicated credits being available. Doing so means that every TLP will contain a Flit Mode Local TLP Prefix.

7. [Merged] for Shared Completion credits indicates that Shared Completions and Shared Posted credits share a common pool of credits. [Merged] is not permitted on Shared Posted or Shared Non-Posted credits. Dedicated credits are never merged.

| Row | Local and Remote ↑↓Extended VC Count↓ ↑↓Extended VC Count↑ | Merged | InitFC Count | Dedicated / Shared | Kind | DataScale and HdrScale | DataFC and HdrFC | Notes |
|-----|--|--------|--------------|--------------------|------|------------------------|------------------|-------|
|-----|--|--------|--------------|--------------------|------|------------------------|------------------|-------|

Use of [Merged] shared credits must be consistent across VCs. If one VC uses [Merged] shared credits, all VCs must also use [Merged] shared credits.

Use of [Merged] shared credits must be consistent between Hdr and Data. If CplH uses [Merged] shared credits, CplD must also use [Merged] shared credits.

When [Merged] is used, and P Hdr shared credits are not [Infinite.3], UpdateFC DLLPs for CplH must use the same scale factor as PH.

When [Merged] is used, and P Data shared credits are not [Infinite.3], UpdateFC DLLPs for CplD must use the same scale factor as PD.

When [Merged], UpdateFC DLLPs and Optimized_Update_FCs return credits as if [Merged] was not enabled (i.e., based on the type of TLP being freed even though there is a single shared credit pool). Doing this provides the transmitter visibility into remote buffer occupancy (Posted vs Completion) and allows it to implement vendor specific mechanisms to manage that occupancy.

When [Merged] is used and P Hdr shared credits are not [Infinite.3], and at least one VC must use non [Zero] P Hdr shared credits.

When [Merged] is used and P Data shared credits are not [Infinite.3], and at least one VC must use non [Zero] P Data shared credits.

- 8. When (Local ≠ 0 and Remote = 0), rows 5-8 are recommended, but rows 11-13 are permitted.
- 9. The Shared / Dedicated mechanism is defined so that shared credits are the common case. TLPs consuming dedicated credits use the Flit Mode Local TLP Prefix and thus consume an additional DW. In rows 3-4, all credits are shared.
- 10. To avoid deadlock, Posted and Completion credits must not be [Zero].

-
- Except as needed to ensure at least the required frequency of InitFC1 DLLP transmission, the Data Link Layer must not block other transmissions.
 - Note that this includes all Physical Layer initiated transmissions (e.g., Ordered Sets), Ack and Nak DLLPs (when applicable), and TLPs using VCs that have previously completed initialization (when applicable)
 - Process received InitFC1 and InitFC2 DLLPs:
 - Record the indicated HdrFC and DataFC values
 - If the Receiver supports Scaled Flow Control, record the indicated HdrScale and DataScale values.
 - Set flag FI1 once FC unit values have been recorded for each of P, NP, and Cpl for VCx.
 - In Non-Flit Mode flag FI1 is Set when the three dedicated FC unit values have been recorded.
 - In Flit Mode, flag FI1 is Set when all six FC unit values have been recorded.
 - Exit to FC_INIT2 if:

- Flag FI1 has been Set indicating that FC unit values have been recorded for each of P, NP, and Cpl for VCx
- Rules for state ***FC_INIT2*** :
 - While in ***FC_INIT2*** :
 - Transaction Layer must block transmission of TLPs using VCx
 - In ~~↓↑non-Flit Mode, ↓↑Non-Flit Mode,~~ transmit the following three InitFC2 DLLPs for VCx in the following relative order:
 - InitFC2-P [Dedicated] (first)
 - InitFC2-NP [Dedicated] (second)
 - InitFC2-Cpl [Dedicated] (third)
 - In Flit Mode, transmit the following six InitFC2 DLLPs for VCx in the following relative order:
 - InitFC2-P [Dedicated] (first)
 - InitFC2-NP [Dedicated] (second)
 - InitFC2-Cpl [Dedicated] (third)
 - InitFC2-P [Shared] (fourth)
 - InitFC2-NP [Shared] (fifth)
 - InitFC2-Cpl [Shared] (sixth)
 - The three (six) InitFC2 DLLPs must be transmitted at least once every 34 µs.
 - Time spent in the Recovery or Configuration LTSSM states does not contribute to this limit.
 - It is strongly ~~↓↑encouraged~~ ↑↑recommended that the InitFC2 DLLP transmissions are repeated frequently, particularly when there are no other TLPs or DLLPs available for transmission.
 - Set DataFC, DataScale, HdrFC, and HdrScale as shown in § [Table 3-2](#) and § [Table 3-3](#)
 - Except as needed to ensure at least the required frequency of InitFC2 DLLP transmission, the Data Link Layer must not block other transmissions
 - Note that this includes all Physical Layer initiated transmissions (for example, Ordered Sets), Ack and Nak DLLPs (when applicable), and TLPs using VCs that have previously completed initialization (when applicable)
 - Process received InitFC1 and InitFC2 DLLPs:
 - Ignore the received HdrFC, HdrScale, DataFC, and DataScale values
 - Set flag FI2 on receipt of any InitFC2 DLLP for VCx
 - Set flag FI2 on receipt of any TLP on VCx, any UpdateFC DLLP for VCx, or, in Flit Mode, any Optimized_Update_FC
 - Signal completion and exit if:
 - Flag FI2 has been Set
 - If Scaled Flow Control is activated on the Link, the Transmitter must send 01b, 10b, or 11b for HdrScale and DataScale in all UpdateFC DLLPs for VCx.
 - If the Scaled Flow Control is not supported or if Scaled Flow Control is not activated on the Link, the Transmitter must send 00b for HdrScale and DataScale in all UpdateFC DLLPs for VCx.

IMPLEMENTATION NOTE: EXAMPLE OF FLOW CONTROL INITIALIZATION §

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

Base 6.4 vs Base 6.3

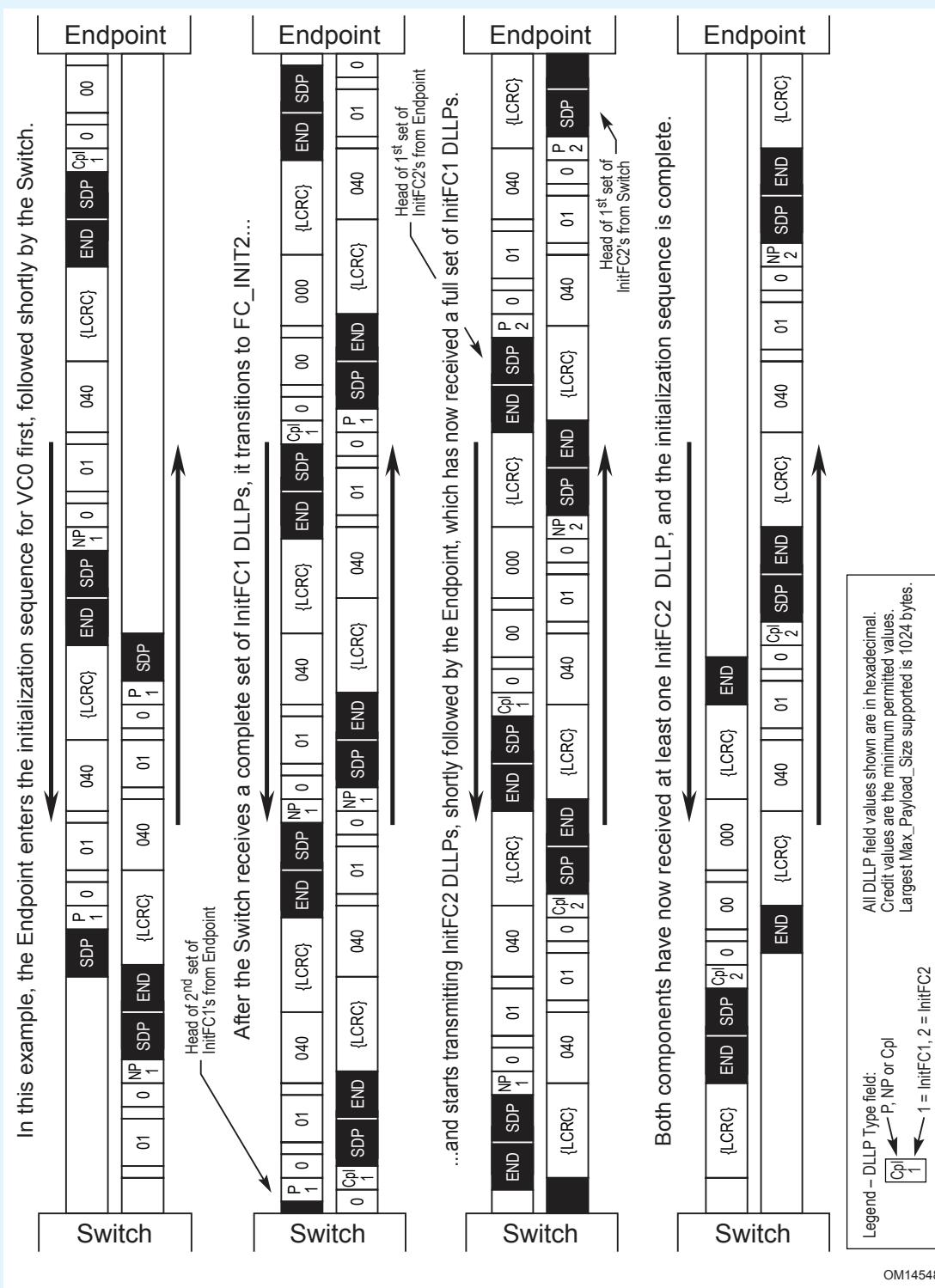


Figure 3-3 VCO Flow Control Initialization Example with 8b/10b Encoding-based Framing ↑- Non-Flit Mode ↑ §

3.4.2 Scaled Flow Control §

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

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

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

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

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

- The InitFC1 and InitFC2 DLLPs that are transmitted must contain 01b, 10b, or 11b in the HdrScale field. The value is determined by the maximum number of header credits that will be outstanding of the indicated credit type as defined in § Table 3-4 .
- The InitFC1 and InitFC2 DLLPs that are transmitted must contain 01b, 10b, or 11b in the DataScale field. The value is determined by the maximum number of data payload credits that will be outstanding of the indicated credit type as defined in § Table 3-4 .

Table 3-4 Scaled Flow Control Scaling Factors §

| Scale Factor | Scaled Flow Control Supported and Activated | Credit Type | Min Credits | Max Credits | Field Width | FC DLLP field | |
|--------------|---|-------------|-------------|-------------|-------------|---------------|-------------|
| | | | | | | Transmitted | Received |
| 00b | No | Hdr | 1 | 127 | 8 bits | HdrFC | HdrFC |
| | | Data | 1 | 2,047 | 12 bits | DataFC | DataFC |
| 01b | Yes | Hdr | 1 | 127 | 8 bits | HdrFC | HdrFC |
| | | Data | 1 | 2,047 | 12 bits | DataFC | DataFC |
| 10b | Yes | Hdr | 4 | 508 | 10 bits | HdrFC >> 2 | HdrFC << 2 |
| | | Data | 4 | 8,188 | 14 bits | DataFC >> 2 | DataFC << 2 |
| 11b | Yes | Hdr | 16 | 2,032 | 12 bits | HdrFC >> 4 | HdrFC << 4 |
| | | Data | 16 | 32,752 | 16 bits | DataFC >> 4 | DataFC << 4 |

3.5 Data Link Layer Packets (DLLPs) §

The following DLLPs are used to support Link operations:

- Data Link Feature DLLP: For negotiation of supported features

- Ack DLLP: TLP Sequence Number acknowledgement; used to indicate successful receipt of some number of TLPs (NFM only)
- Nak DLLP: TLP Sequence Number negative acknowledgement; used to initiate a Data Link Layer Retry (NFM only)
- InitFC1, InitFC2, and UpdateFC DLLPs; used for Flow Control
- DLLPs used for Power Management
- DLLPs used for Link Management (L0p)

3.5.1 Data Link Layer Packet Rules §

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

In Non-Flit Mode, all DLLPs include the following fields:

- DLLP Type - Specifies the type of DLLP. The defined encodings are shown in § [Table 3-5](#).
- 24 bits of DLLP Type specific information
- 16-bit CRC

In Flit Mode, DLLPs are transmitted in the DLP bytes of a Flit. They consist of:

- DLLP Type - Specifies the type of DLLP. The defined encodings are shown in § [Table 3-5](#).
- 24 bits of DLLP Type specific information

See § [Figure 3-4](#) and § [Figure 3-5](#) below.

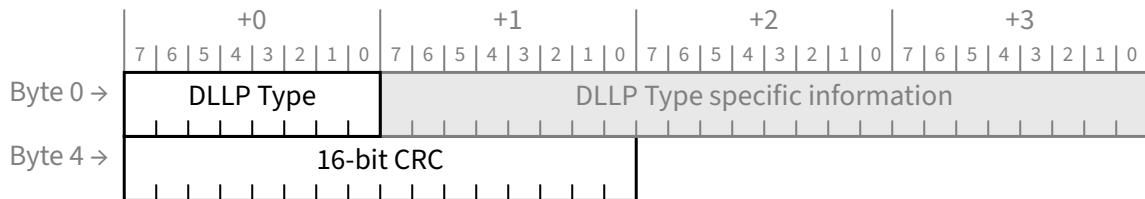


Figure 3-4 DLLP Type and CRC Fields (Non-Flit Mode) §

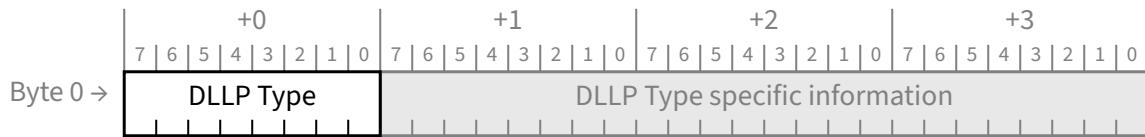


Figure 3-5 DLLP Type Field (Flit Mode) §

Base 6.4 vs Base 6.3

Table 3-5 DLLP Type Encodings §

| Encodings (b) ↑↓↑(Note 4)↑ | DLLP Type (Note 3) | References |
|--|---|--|
| 0000 0000 | Ack (Non-Flit Node) | § Figure 3-6 ↑↓, Insert text reference here↓ |
| | NOP2 (Flit Mode) | § Figure 3-8 ↑↓, Insert text reference here↓ |
| 0000 0001 | MRInit (Non-Flit Mode) (Deprecated) | See [MR-IOV] Note 1 |
| | Reserved (Flit Mode) | |
| 0000 0010 | Data Link Feature | § Figure 3-14 ↑↓, Insert text reference here↓ |
| 0000 0011 | Alternate Protocol Use 1 | Not used by PCI Express. Only permitted after an Alternate Protocol has been negotiated (see § Section 4.2.5.2). Meaning is Alternate Protocol specific. For example, see [CXL-3.0]. |
| 0000 0100 | Alternate Protocol Use 2 | |
| 0001 0000 | Nak (Non-Flit Mode) | § Figure 3-6 ↑↓, Insert text reference here↓ |
| | Reserved (Flit Mode) | |
| 0010 0000 | PM_Enter_L1 | § Figure 3-12 , § Section 5.3.2.1 |
| 0010 0001 | PM_Enter_L23 | § Figure 3-12 , § Section 5.3.2.3 |
| 0010 0011 | PM_Active_State_Request_L1 | § Figure 3-12 , § Section 5.4.1.3.1 |
| 0010 0100 | PM_Request_Ack | § Figure 3-12 , § Section 5.4.1.3.1 , § Section 5.3.2.1 |
| 0010 1000 | Reserved (Non-Flit Mode) | |
| | Link Management (Flit Mode) | § Figure 3-15 , § Table 4-56 |
| 0011 0000 | Vendor-Specific | § Figure 3-13 ↑↓, Insert text reference here↓ |
| 0011 0001 | NOP | § Figure 3-7 ↑↓, Insert text reference here↓ |
| 0100 0v ₂ v ₁ v ₀ 0100 1v ₂ v ₁ v ₀ | InitFC1-P ↑↓(v[2:0] specifies Virtual Channel)↓ | § Figure 3-9 ↑↓, Insert text reference here↓ |
| 0101 0v ₂ v ₁ v ₀ 0101 1v ₂ v ₁ v ₀ | InitFC1-NP | |
| 0110 0v ₂ v ₁ v ₀ 0110 1v ₂ v ₁ v ₀ | InitFC1-Cpl | |
| 0111 0v ₂ v ₁ v ₀ | MRInitFC1 ↑↓(Non-Flit Mode)↓ ↑↓(Non-Flit Mode)↑ (Deprecated) | See [MR-IOV] Note 2 |

Base 6.4 vs Base 6.3

| Encodings (b) ↑↓↑(Note 4)↑ | DLLP Type (Note 3) | References |
|--|---|---|
| | Reserved (Flit Mode) | |
| 1100 0v ₂ v ₁ v ₀ 1100 1v ₂ v ₁ v ₀ | InitFC2-P | § Figure 3-10 ↑↓, Insert text reference here↓ |
| 1101 0v ₂ v ₁ v ₀ 1101 1v ₂ v ₁ v ₀ | InitFC2-NP | |
| 1110 0v ₂ v ₁ v ₀ 1110 1v ₂ v ₁ v ₀ | InitFC2-Cpl | |
| 1111 0v ₂ v ₁ v ₀ | MRInitFC2 ↑↓(Non-Flit Mode)↑ ↑↓(Non-Flit Mode)↑ (Deprecated) | See [MR-IOV] Note 2 |
| | Reserved (Flit Mode) | |
| 1000 0v ₂ v ₁ v ₀ 1000 1v ₂ v ₁ v ₀ | UpdateFC-P | § Figure 3-11 ↑↓, Insert text reference here↓ |
| 1001 0v ₂ v ₁ v ₀ 1001 1v ₂ v ₁ v ₀ | UpdateFC-NP | |
| 1010 0v ₂ v ₁ v ₀ 1010 1v ₂ v ₁ v ₀ | UpdateFC-Cpl | |
| 1011 0v ₂ v ₁ v ₀ | MRUpdateFC (Non-Flit Mode) (Deprecated) | See [MR-IOV] Note 2 |
| | Reserved (Flit Mode) | |
| All other encodings | Reserved | |

↑↓Notes:↑

1. The deprecated MR-IOV protocol uses this encoding for the MRInit negotiation. The MR-IOV protocol assumes that non-MR-IOV components will silently ignore these DLLPs.
2. The deprecated MR-IOV protocol uses these encodings only after the successful completion of MRInit negotiation.

| Encodings (b) ↑↓↑(Note 4)↑ | DLLP Type (Note 3) | References |
|-------------------------------------|--------------------|------------|
|-------------------------------------|--------------------|------------|

3. Received DLLPs not supported by the Receiver are silently ignored. See § Section 3.6.2.2 and § Section 3.6.2.3 .
4. ↑↓↑v↑↑↓↑2↑↑↓↑v↑↑↓↑1↑↑↓↑0↑↑↓↑specifies Virtual Channel↑

In § Figure 3-6 through § Figure 3-15 the 16-bit CRC is not shown. In Non-Flit Mode, the CRC is present as shown in § Figure 3-4 .

- For Ack and Nak DLLPs (see § Figure 3-6):
 - The AckNak_Seq_Num field is used to indicate what TLPs are affected
 - Transmission and reception is handled by the Data Link Layer according to the rules provided in § Section 3.6 .
 - These DLLPs are not used in Flit Mode. In Flit Mode, the Ack encoding is used for the NOP2 .
- 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 data payload of the indicated type (P, NP, or Cpl).
 - When Scaled Flow Control is supported, the HdrScale field contains the scaling factor for headers of the indicated type. Encodings are defined in § Table 3-6 .
 - When Scaled Flow Control is supported, the DataScale field contains the scaling factor for data payload of the indicated type. Encodings are defined in § Table 3-6 .
 - When Scaled Flow Control is not supported, the HdrScale and ↑↓Data Scale↓↑↑DataScale↑ fields are Reserved.
 - If Scaled Flow Control is activated, the HdrScale and DataScale fields must be set to 01b, 10b, or 11b in all InitFC1, InitFC2, and UpdateFC DLLPs transmitted.
 - In UpdateFCs, a Transmitter is only permitted to send non-zero values in the HdrScale and DataScale fields if it supports Scaled Flow Control, Scaled Flow control is activated, and it received non-zero values for HdrScale and DataScale in the InitFC1s and InitFC2s it received for this VC.

In Flit Mode, the Optimized_Update_FC mechanism is supported in addition to the UpdateFC DLLP. Optimized_Update_FCs do not contain HdrScale and DataScale fields, so the Transmitter and Receiver must treat the HdrFC and DataFC fields using corresponding HdrScale and DataScale fields advertised during initialization. For debug as well as ease of use by debug tools such as Logic Analyzers, devices must send at least one DLLP every 10 µs with an UpdateFC DLLP per VC with the scaled credit information. It is strongly recommended that a Transmitter cycles through the VCs with finite non-0 credits in the Optimized_Update_FC as long as there is a credit to be released in the corresponding VC.

 - The packet formats are shown in § Figure 3-9 , § Figure 3-10 , and § Figure 3-11 .
 - Transmission is triggered by the Data Link Layer when initializing Flow Control for a Virtual Channel (see § Section 3.4), and following Flow Control initialization by the Transaction Layer according to the rules in § Section 2.6 .
 - Checked for integrity on reception by the Data Link Layer and if correct, the information content of the DLLP is passed to the Transaction Layer. If the check fails, the information must be discarded.

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

Table 3-6 HdrScale and DataScale Encodings §

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

- For Power Management (PM) DLLPs (see § Figure 3-12):
 - Transmission is triggered by the component's power management logic according to the rules in § Chapter 5.
 - Checked for integrity on reception by the Data Link Layer, then passed to the component's power management logic
- For Vendor-Specific DLLPs (see § Figure 3-13)
 - It is recommended that receivers silently ignore Vendor-Specific DLLPs unless enabled by implementation specific mechanisms.
 - It is recommended that transmitters not send Vendor-Specific DLLPs unless enabled by implementation specific mechanisms.
- For NOP DLLPs (see § Figure 3-7) and NOP2 DLLPs (see § Figure 3-8).
 - Receivers shall discard this DLLP without action, unless otherwise specified, after checking it for data integrity.⁶³
- For Data Link Feature DLLPs (see § Figure 3-14)
 - The Feature Ack bit is Set to 1b to indicate that the transmitting Port has received a Data Link Feature DLLP.
 - The Feature Supported field indicates the Data Link Features supported and/or attribute values for the transmitting Port. The individual bits of this field are defined in § Table 3-1 .
- For Link Management DLLPs (see § Figure 3-15)
 - In Non-Flit Mode, receivers shall discard this DLLP without action after checking it for data integrity.⁶⁴

63. This is a special case of the more general rule for unsupported DLLP Type encodings (see § Section 3.6.2.2 and § Section 3.6.2.3).

64. This is a special case of the more general rule for unsupported DLLP Type encodings (see § Section 3.6.2.2 and § Section 3.6.2.3).

Base 6.4 vs Base 6.3

```
↑↓[{"debug": false, "preClass": "hide", "forceFit": false, "width": 32, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [{"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "0000 0000 Ack", "value": ["0", "0", "0", "x", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "0001 0000 Nak", "value": ["0", "0", "0", "x", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 3, "name": "AckNak_Seq_Num", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}]]}↓
```

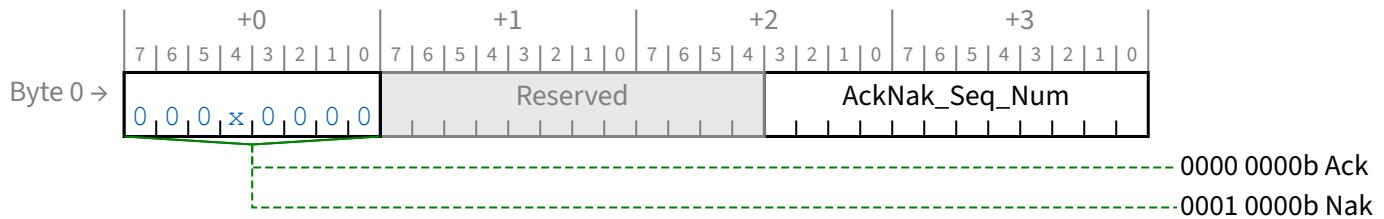


Figure 3-6 Data Link Layer Packet Format for Ack and Nak (Non-Flit Mode)

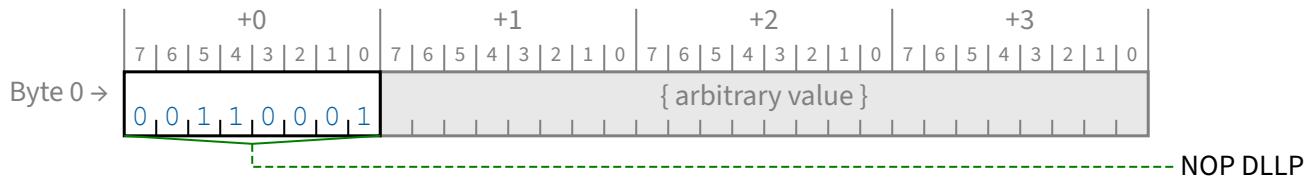


Figure 3-7 Data Link Layer Packet Format for NOP

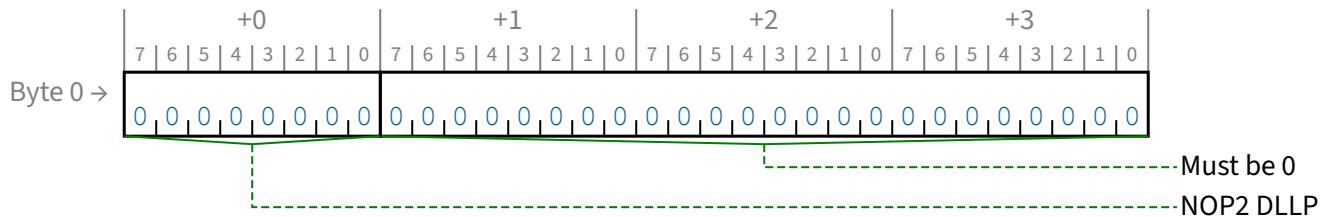


Figure 3-8 Data Link Layer Packet Format for NOP2 (Flit Mode)

```
↑↓[{"debug": false, "preClass": "hide", "forceFit": false, "width": 32, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [{"lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "0100 InitFC1 P", "value": ["0", "1", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "0101 InitFC1 NP", "value": ["0", "1", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "0110 InitFC1 Cpl", "value": ["0", "1", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 3, "msbyte": 0, "msbit": 3, "name": "0 Shared Flow Control", "value": "x", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 3, "msbyte": 0, "msbit": 3, "name": "1 Dedicated Flow Control", "value": "x", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 2, "msbyte": 0, "msbit": 0, "name": "VC[2:0]", "value": "00", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 7, "name": "Flit Mode: 00b - Infinite Credits if HdrFC = 00h", "value": "00", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 7, "name": "Flit Mode: 01b - Zero Credits when HdrFC = 00h", "value": "01", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 7, "name": "Flit Mode: 10b - Merged Shared Credits if HdrFC = 00h", "value": "10", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 6, "msbyte": 1, "msbit": 6, "name": "HdrFC", "value": "00", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Data Scale", "value": "00", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Flit Mode: 00b", "value": "00", "addClass": "regFieldVerySmallText", "attr": "ro"}]]}↓
```

Infinite Credits when DataFC = 000h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Flit Mode: 01b - Zero Credits when DataFC = 000h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Flit Mode: 10b - Merged Shared Credits when DataFC = 000h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 3, "name": "DataFC", "attr": "ro" }]]]

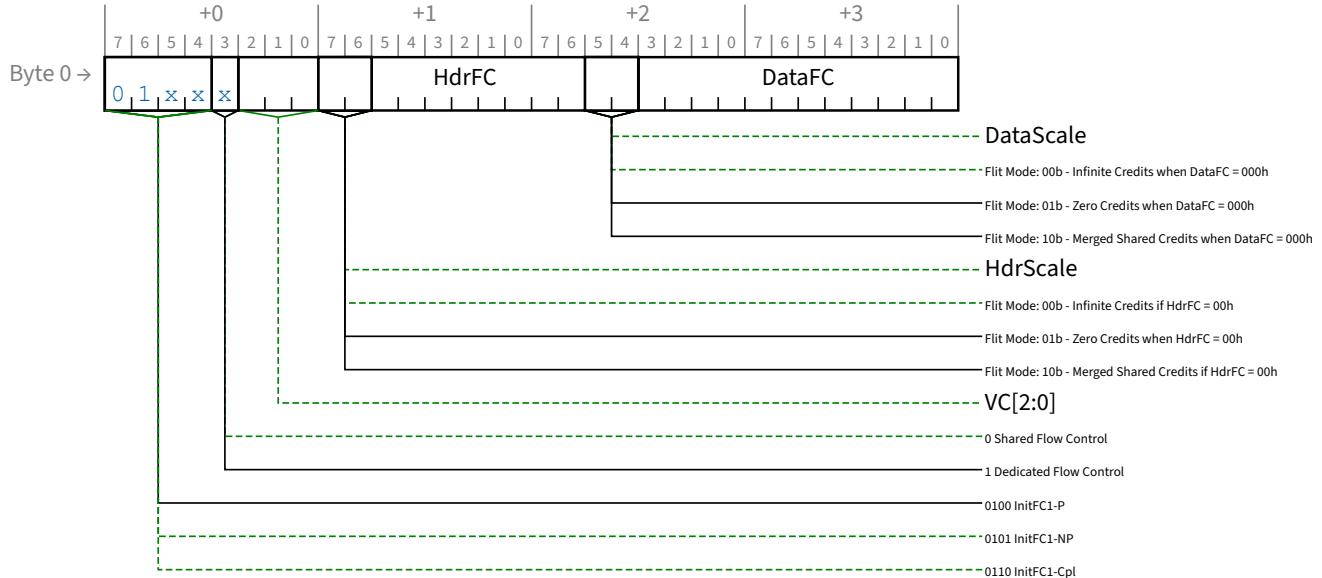


Figure 3-9 Data Link Layer Packet Format for InitFC1 §

See also § Table 3-2 and § Table 3-3 . Note: In Base 6.0.1, the encoding of byte 0, bit 3 in § Figure 3-9 was updated to match § Table 3-3 .

↑↑ { "debug": false, "preClass": "hide", "forceFit": false, "width": 32, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [{ "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "1100 InitFC2_P", "value": ["1", "1", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "1101 InitFC2_NP", "value": ["1", "1", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "1110 InitFC2_Cpl", "value": ["1", "1", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 3, "msbyte": 0, "msbit": 3, "name": "0 Shared Flow Control", "value": "x", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 3, "msbyte": 0, "msbit": 3, "name": "1 Dedicated Flow Control", "value": "x", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 2, "msbyte": 0, "msbit": 0, "name": "VC[2:0]", "value": "ro", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 7, "name": "Flit Mode: 00b - Infinite Credits if HdrFC = 00h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 7, "name": "Flit Mode: 01b - Zero Credits when HdrFC = 00h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 6, "msbyte": 1, "msbit": 7, "name": "Flit Mode: 10b - Merged Shared Credits if HdrFC = 00h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 1, "msbit": 6, "name": "HdrFC", "value": "ro", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Data Scale", "value": "ro", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Flit Mode: 00b - Infinite Credits when DataFC = 000h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Flit Mode: 01b - Zero Credits when DataFC = 000h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 5, "name": "Flit Mode: 10b - Merged Shared Credits when DataFC = 000h", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 3, "name": "DataFC", "value": "ro", "addClass": "regFieldVerySmallText", "attr": "ro" }]]]

Base 6.4 vs Base 6.3

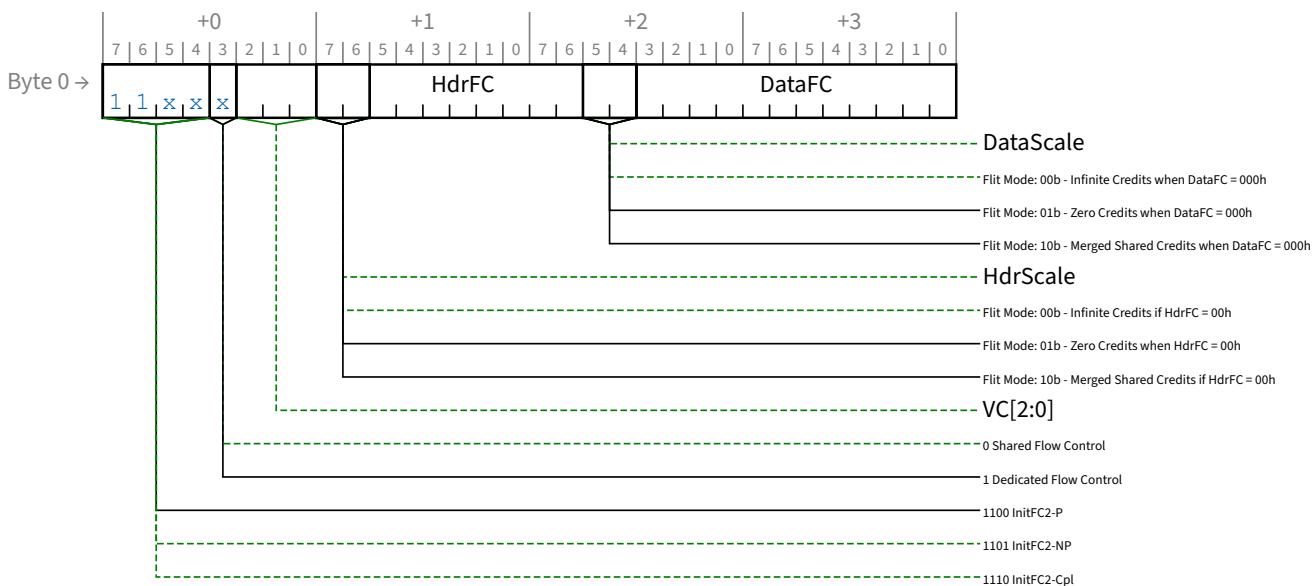


Figure 3-10 Data Link Layer Packet Format for InitFC2 §

See also § Table 3-2 and § Table 3-3 . Note: In Base 6.0.1, the encoding of byte 0, bit 3 in § Figure 3-10 was updated to match § Table 3-3 .

```
↑\v{ "debug": false, "preClass": "hide", "forceFit": false, "width": 32, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [ { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "1000 UpdateFC2_P", "value": ["1", "0", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "1001 UpdateFC2_NP", "value": ["1", "0", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "1010 UpdateFC2_Cpl", "value": ["1", "0", "x", "x"], "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 3, "msbyte": 0, "msbit": 3, "name": "0 Shared Flow Control", "value": "x", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 3, "msbyte": 0, "msbit": 3, "name": "1 Dedicated Flow Control", "value": "x", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 2, "msbyte": 0, "msbit": 7, "name": "Hdr Scale", "value": "ro" }, { "lsbyte": 2, "lsbit": 6, "msbyte": 1, "msbit": 6, "name": "HdrFC", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 3, "name": "DataFC", "attr": "ro" } ] } }↑
```

Base 6.4 vs Base 6.3

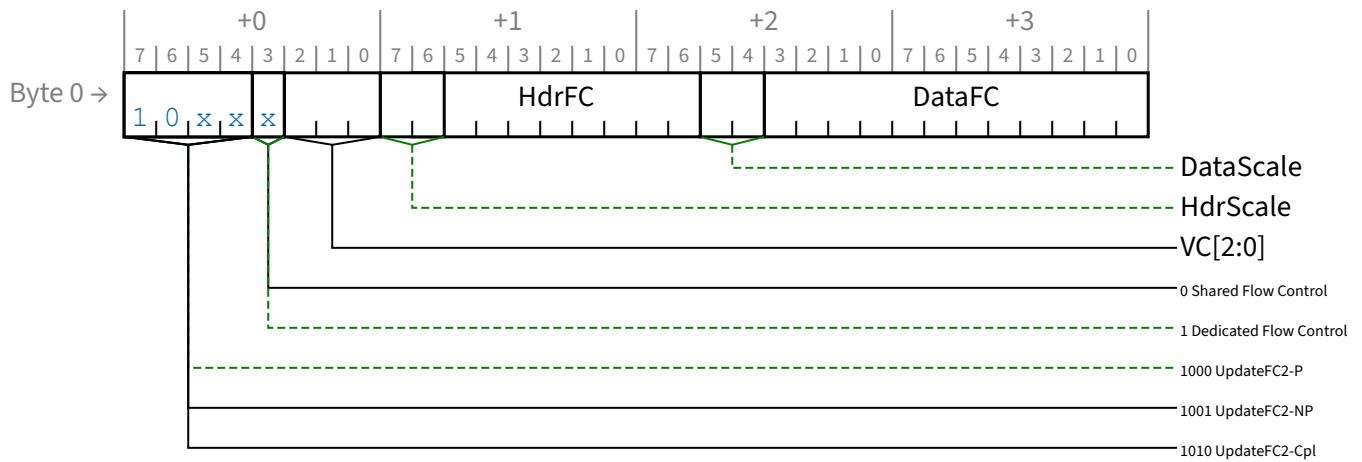


Figure 3-11 Data Link Layer Packet Format for UpdateFC §

Note: In Base 6.0.1, the encoding of byte 0, bit 3 in § Figure 3-11 was updated to match § Table 3-3 .

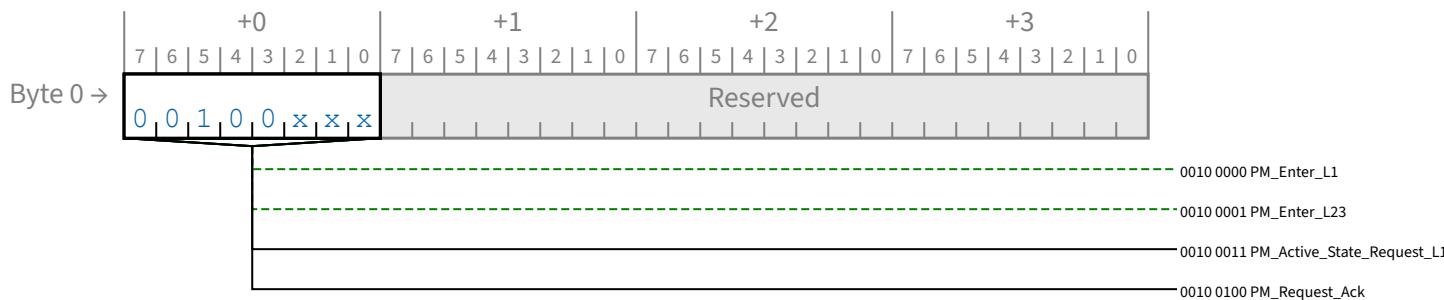


Figure 3-12 Data Link Layer Packet Format for Power Management §

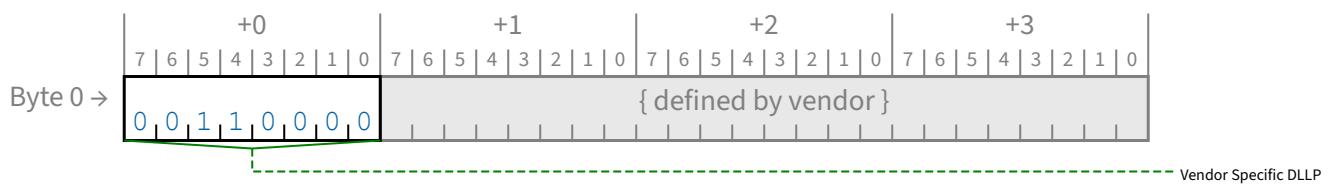


Figure 3-13 Data Link Layer Packet Format for Vendor-Specific §

Base 6.4 vs Base 6.3

```
↓↓[{"debug": false, "preClass": "hide", "forceFit": false, "width": 32, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [{"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Data Link Feature DLLP", "value": ["0", "0", "0", "0", "0", "1", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 7, "name": "Feature Ack", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 1, "msbit": 6, "name": "Feature Support", "addClass": "regFieldVerySmallText", "attr": "ro"}]]↓
```

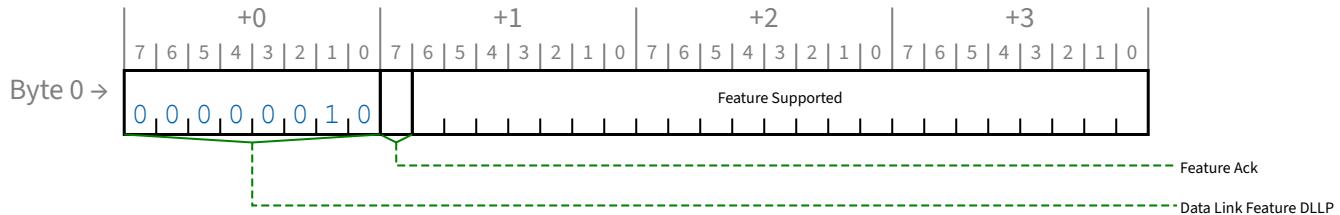


Figure 3-14 Data Link Layer Packet Format for [Data Link Feature DLLP](#) [Data Link Feature DLLP](#) §

```
↓↓[{"debug": false, "preClass": "hide", "forceFit": false, "width": 32, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [{"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "Link Management DLLP", "value": ["0", "0", "1", "0", "0", "1", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "Link Mgmt Type = L0p", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 5, "msbyte": 2, "msbit": 7, "name": "Reserved", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 4, "name": "L0p.Priority", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 2, "lsbit": 0, "msbyte": 2, "msbit": 3, "name": "L0p.Cmd", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 4, "msbyte": 3, "msbit": 7, "name": "Response Payload", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 3, "msbit": 3, "name": "Link Width", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "addClass": "regFieldVerySmallText", "attr": "ro"}]]↓
```

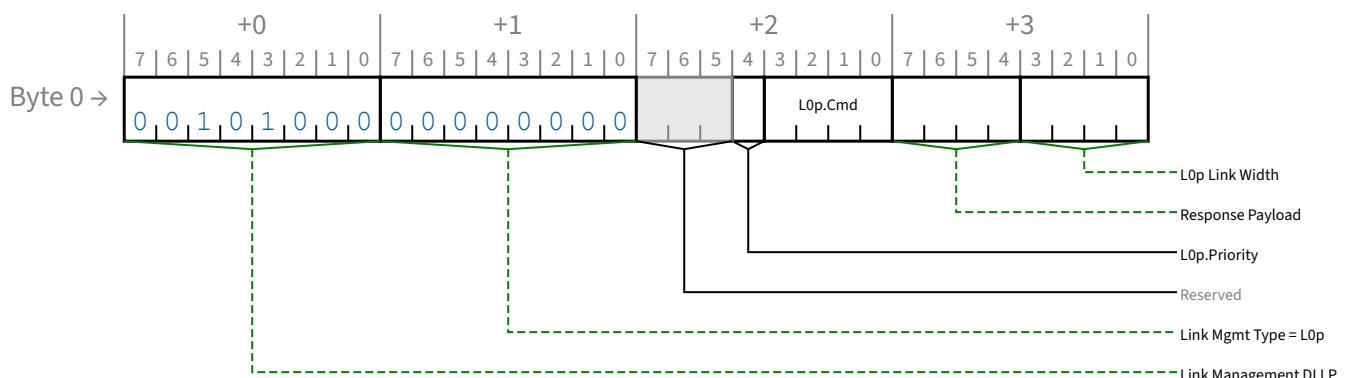


Figure 3-15 Data Link Packet Layer Format for [Link Management](#) [Link Management](#) (Flit Mode) §

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

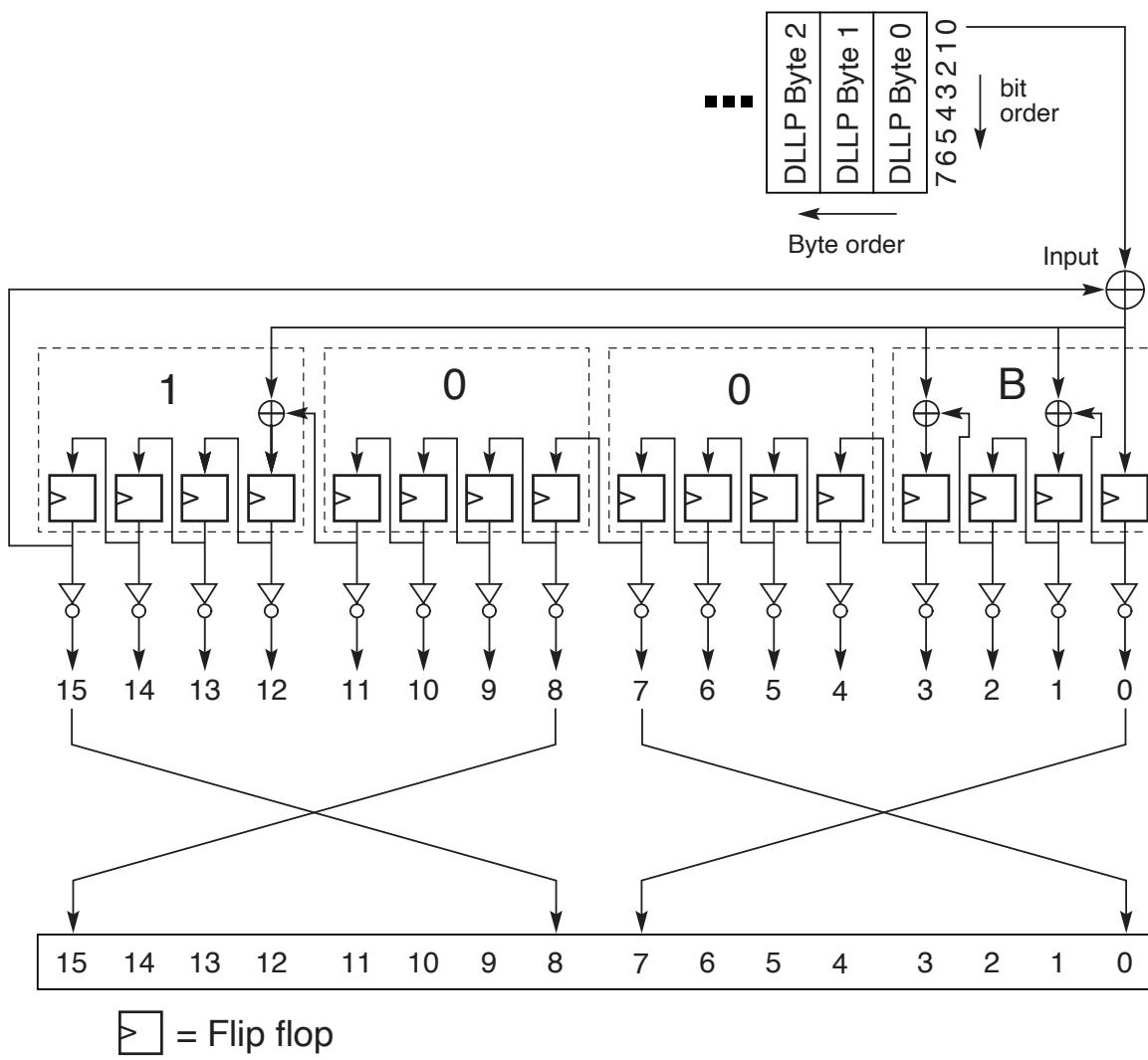
- DLLPs are differentiated from TLPs when they are presented to, or received from, the Physical Layer.
- DLLP data integrity is protected using a 16-bit CRC (NFM only)
- The CRC value is calculated using the following rules (see § Figure 3-16):
 - The polynomial used for CRC calculation has a coefficient expressed as 100Bh
 - The seed value (initial value for CRC storage registers) is FFFFh
 - CRC calculation starts with bit 0 of byte 0 and proceeds from bit 0 to bit 7 of each byte

- Note that CRC calculation uses all bits of the DLLP, regardless of field type, including Reserved fields. The result of the calculation is complemented, then placed into the 16-bit CRC field of the DLLP as shown in § Table 3-7 .

Table 3-7 Mapping of Bits into CRC Field §

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

Base 6.4 vs Base 6.3



OM13785

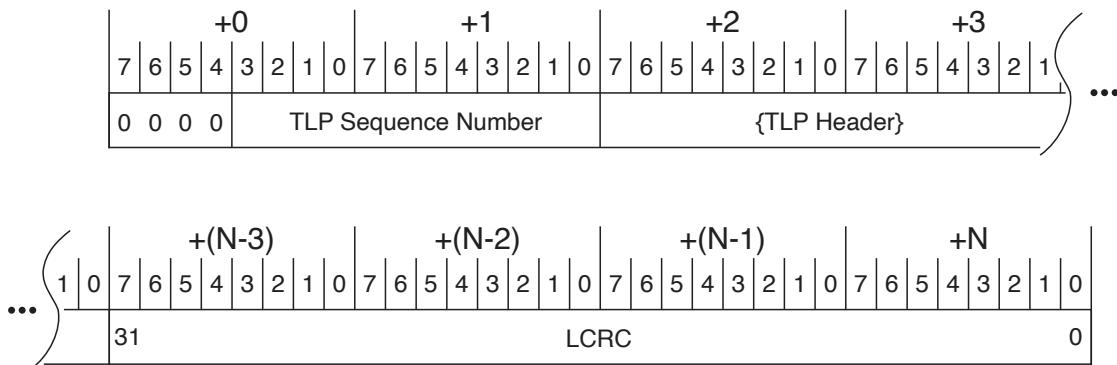
Figure 3-16 Diagram of CRC Calculation for DLLPs §

3.6 Data Integrity Mechanisms §

3.6.1 Introduction §

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

The format of a TLP with the TLP Sequence Number and LCRC ↓↓code↓↓ applied is shown in § Figure 3-17 .



OM13786A

Figure 3-17 TLP with LCRC and TLP Sequence Number Applied - Non-Fit Mode §

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

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

3.6.2 LCRC, $\uparrow\downarrow$ TLP \uparrow 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 will instruct the Physical Layer to retrain the Link (via the LTSSM Recovery state, § Section 4.2.7). 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 $\uparrow\downarrow$ TLP \uparrow Sequence Number and to support Data Link Layer Retry are described in terms of conceptual counters and flags. This description does not imply nor require a particular implementation and is used only to clarify the requirements.

3.6.2.1 LCRC and $\uparrow\downarrow$ TLP \uparrow Sequence Number Rules (TLP Transmitter) §

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

- The following 12-bit counters are used:
 - NEXT_TRANSMIT_SEQ - Stores the packet sequence number applied to TLPs
 - Set to 000h in DL_Inactive state
 - ACKD_SEQ - Stores the sequence number acknowledged in the most recently received Ack or Nak DLLP.
 - Set to FFFh in DL_Inactive state
- The following 3-bit counter is used:
 - REPLAY_NUM - Counts the number of times the Retry Buffer has been re-transmitted

- Set to 000b in DL_Inactive state
- The following timer is used:
 - **REPLAY_TIMER** - Counts time that determines when a replay is required, according to the following rules:
 - Started at the last Symbol of any TLP transmission or retransmission, if not already running
 - For each replay, reset and restart REPLAY_TIMER when sending the last Symbol of the first TLP to be retransmitted
 - Resets and restarts for each Ack DLLP received while there are more unacknowledged TLPs outstanding, if, and only if, the received Ack DLLP acknowledges some TLP in the retry buffer.
 - Note: This ensures that REPLAY_TIMER is reset only when forward progress is being made
 - Reset and hold until restart conditions are met for each Nak received (except during a replay) or when the REPLAY_TIMER expires
 - Not advanced during Link retraining (holds its value when the LTSSM is in the Recovery or Configuration state). Refer to § Section 4.2.6.3 and § Section 4.2.6.4.
 - If Protocol Multiplexing is supported, optionally not advanced during the reception of PMUX Packets (see § Appendix G.).
 - Resets and holds when there are no outstanding unacknowledged TLPs

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

- The Transaction Layer indicates the start and end of the TLP to the Data Link Layer while transferring the TLP
 - The Data Link Layer treats the TLP as a “black box” and does not process or modify the contents of the TLP
- Each TLP is assigned a 12-bit sequence number when it is accepted from the Transmit side of the Transaction Layer
 - Upon acceptance of the TLP from the Transaction Layer, the packet sequence number is applied to the TLP by:
 - prepending the 12-bit value in NEXT_TRANSMIT_SEQ to the TLP
 - prepending 4 bits to the TLP, preceding the ↓↓sequence number↓↓↑↑TLP Sequence Number↑↑ (see § Figure 3-18)
 - If the equation:

$$(\text{NEXT_TRANSMIT_SEQ} - \text{ACKD_SEQ}) \bmod 4096 \geq 2048$$

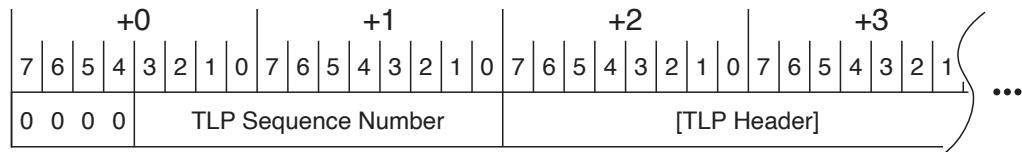
Equation 3-1 Tx SEQ Stall §

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

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

$$\text{NEXT_TRANSMIT_SEQ} = (\text{NEXT_TRANSMIT_SEQ} + 1) \bmod 4096$$

Equation 3-2 Tx SEQ Update §



OM13787A

Figure 3-18 TLP Following Application of TLP Sequence Number and 4 Bits §

Base 6.4 vs Base 6.3

- TLP data integrity is protected during transfer between Data Link Layers using a 32-bit LCRC
- The LCRC value is calculated using the following mechanism (see § Figure 3-19):
 - 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 3-18)
 - LCRC calculation starts with bit 0 of byte 0 (bit 8 of the TLP sequence number) and proceeds from bit 0 to bit 7 of each successive byte.
 - Note that LCRC calculation uses all bits of the TLP, regardless of field type, including Reserved fields
 - The remainder of the LCRC calculation is complemented, and the complemented result bits are mapped into the 32-bit LCRC field as shown in § Table 3-8 .

Table 3-8 Mapping of Bits into LCRC Field §

| LCRC Result Bit | Corresponding Bit Position in the 32-Bit LCRC Field |
|-----------------|---|
| 0 | 7 |
| 1 | 6 |
| 2 | 5 |
| 3 | 4 |
| 4 | 3 |
| 5 | 2 |
| 6 | 1 |
| 7 | 0 |
| 8 | 15 |
| 9 | 14 |
| 10 | 13 |
| 11 | 12 |

Base 6.4 vs Base 6.3

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

Base 6.4 vs Base 6.3

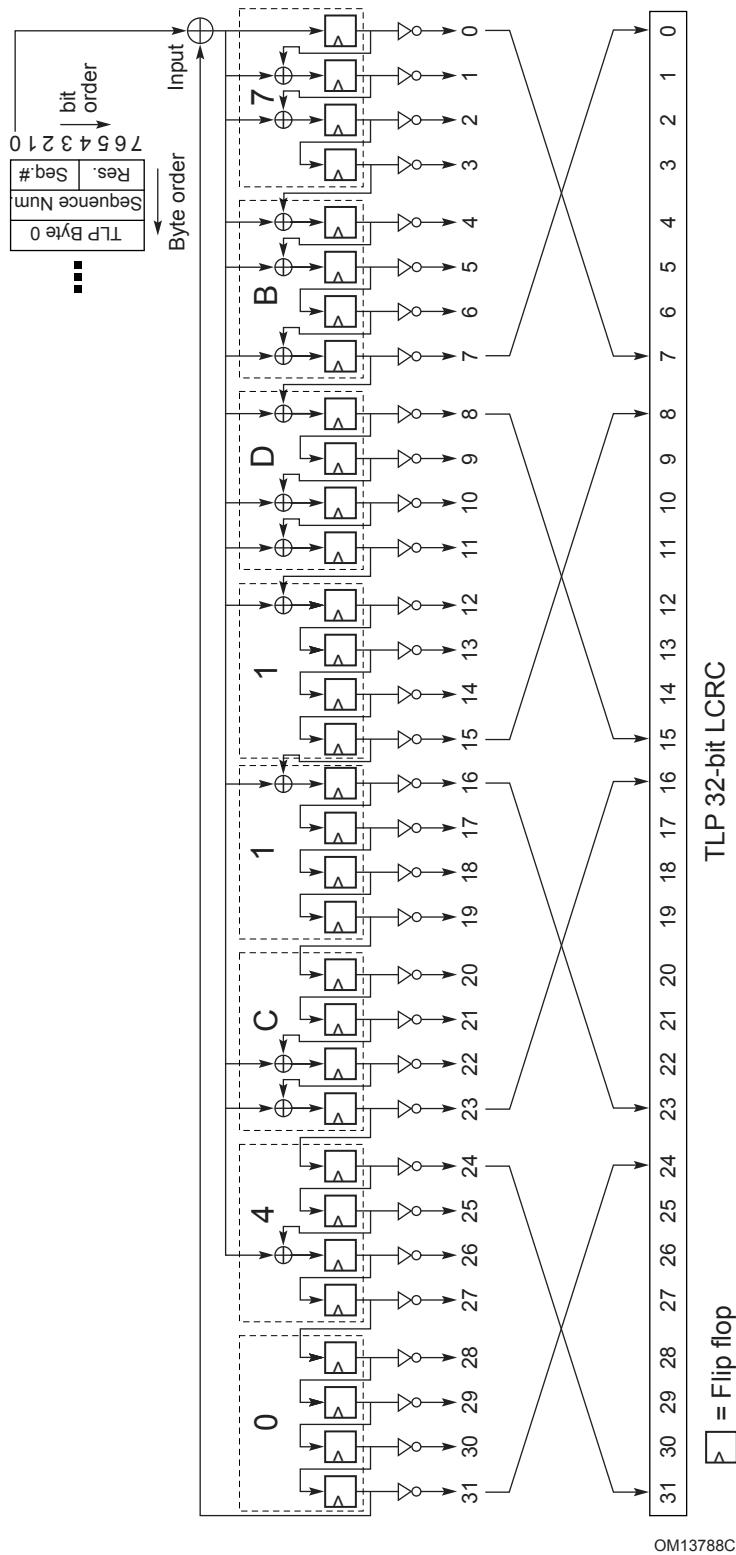


Figure 3-19 Calculation of LCRC §

The 32-bit LCRC field is appended to the TLP following the bytes received from the Transaction Layer (see § Figure 3-17).

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 that will robustly prevent misinterpretation or corruption, the Transmitter must do the following:
 - Transmit all DWs of the TLP when the Physical Layer is using 128b/130b encoding (see § [Section 4.2.2.3.1](#))
 - Use the remainder of the calculated LCRC value without inversion (the logical inverse of the value normally used)
 - Indicate to the Transmit Physical Layer that the TLP is nullified
- When this is done, the Transmitter does not increment NEXT_TRANSMIT_SEQ

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

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

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

- If all TLPs transmitted have been acknowledged (the Retry Buffer is empty), terminate replay, otherwise continue.
Note: In Flit Mode, Replay occurs at the Flit Level. See § [Section 4.2.3.4.2.1.7](#).
- Increment REPLAY_NUM by 2 when operating in Non-Flit Mode. If the Data Rate is 32.0 GT/s or lower, increment REPLAY_NUM by 2. When the replay is initiated by the reception of a Nak that acknowledged some [↑↓TLPs↑](#) in the retry buffer, REPLAY_NUM is reset. It is then permitted (but not required) to be incremented.
 - If REPLAY_NUM rolls over from 110b or 111b to either 000b or 001b, the Transmitter signals the Physical Layer to retrain the Link, and waits for the completion of retraining before proceeding with the replay. This is a reported error associated with the Port (see § [Section 6.2](#)).

Note that Data Link Layer state, including the contents of the Retry Buffer, are not reset by this action unless the Physical Layer reports Physical LinkUp = 0b (causing the Data Link Control and Management State Machine to transition to the [DL_Inactive](#) state).

- If REPLAY_NUM does not roll over from 110b or 111b to either 000b or 001b, continue with the replay.
- Block acceptance of new TLPs from the Transmit Transaction Layer.
- Complete transmission of any TLP currently being transmitted.
- Retransmit unacknowledged TLPs, starting with the oldest unacknowledged TLP and continuing in original transmission order
 - Reset and restart [REPLAY_TIMER](#) when sending the last Symbol of the first TLP to be retransmitted
 - Once all unacknowledged TLPs have been re-transmitted, return to normal operation.
 - If any Ack or Nak DLLPs are received during a replay, the Transmitter is permitted to complete the replay without regard to the Ack or Nak DLLP(s), or to skip retransmission of any newly acknowledged TLPs.
 - Once the Transmitter has started to resend a TLP, it must complete transmission of that TLP in all cases.
 - Ack and Nak DLLPs received during a replay must be processed, and may be collapsed
 - Example: If multiple Acks are received, only the one specifying the latest [↑↓Sequence Number](#) [↑↓sequence number↑](#) value must be considered - Acks specifying earlier

Sequence Number **sequence number** values are effectively “collapsed” into this one

- Example: During a replay, Nak is received, followed by an Ack specifying a later Sequence Number - the Ack supersedes the Nak, and the Nak is ignored.
- Note: Since all entries in the Retry Buffer have already been allocated space in the Receiver by the Transmitter's Flow Control gating logic, no further flow control synchronization is necessary.

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

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

- If the Transmit Retry Buffer contains TLPs for which no Ack or Nak DLLP has been received, and (as indicated by REPLAY_TIMER) no Ack or Nak DLLP has been received for a period exceeding the applicable REPLAY_TIMER Limit, the Transmitter initiates a replay.
 - Simplified REPLAY_TIMER Limits are:
 - A value from 24,000 to 31,000 (inclusive) Symbol Times (-0%/+0%) when the Extended Synch bit is Clear.
 - A value from 80,000 to 100,000 (inclusive) Symbol Times (-0%/+0%) when the Extended Synch bit is Set.
 - If the Extended Synch bit changes state while unacknowledged TLPs are outstanding, implementations are permitted to adjust their REPLAY_TIMER Limit when the Extended Synch bit changes state or the next time the REPLAY_TIMER is reset.
 - Implementations that support 16.0 GT/s or higher data rates must use the Simplified REPLAY_TIMER Limits for operation at all data rates when operating in Non-Flit Mode.
 - Implementations that only support data rates less than 16.0 GT/s are strongly recommended to use the Simplified REPLAY_TIMER Limits for operation at all data rates when operating in Non-Flit Mode, but they are permitted to use the REPLAY_TIMER Limits described in the [PCIe-3.1].
 - The Replay Timeout rules defined in Replay Schedule Rule 0 in § Section 4.2.3.4.2.1.6 must be used for operation at all data rates in Flit Mode.

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

IMPLEMENTATION NOTE: DETERMINING REPLAY_TIMER LIMIT VALUES §

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

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

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

IMPLEMENTATION NOTE: RECOMMENDED PRIORITY OF SCHEDULED TRANSMISSIONS §

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

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

In Flit Mode, DLP information is contained in every Flit and can contain either a DLLP, Optimized_Update_FC, or a Flit_Marker. Currently defined Flit_Markers are related to a specific TLP and have the highest priority. Future Flit_Markers may have lower priorities.

| Recommended Priority | Non-Flit Mode | Flit Mode |
|----------------------|--|--|
| 1 | Completion of any transmission (TLP or DLLP) currently in progress (highest priority). | <i>n/a: DLP information is independent of TLPs</i> |
| 2 | <i>n/a: Poison and Nullify use TLP Framing</i> | Poisoned TLP and Nullified TLP Flit_Markers |
| 3 | Nak DLLP Transmissions | <i>n/a: Ack and Nak use dedicated DLP Symbols, not DLLPs</i> |
| 4 | Ack DLLP transmissions scheduled for transmission as soon as possible due to: receipt of a duplicate TLP -OR- expiration of the Ack latency timer (see § Section 3.6.3.11). | |
| 5 | Flow Control required to satisfy § Section 2.6 : UpdateFC DLLPs | Flow Control required to satisfy § Section 2.6 : UpdateFC DLLPs and/or Optimized_Update_FC |
| 6 | Retry Buffer re-transmissions | <i>n/a: DLP information is independent of TLPs</i> |
| 7 | TLPs from the Transaction Layer | |
| 8 | Flow Control other than that required to satisfy § Section 2.6 : UpdateFC DLLPs | Flow Control other than that required to satisfy § Section 2.6 : Optimized_Update_FC and/or UpdateFC DLLPs |
| 9 | All other DLLP transmissions (lowest priority) | |

3.6.2.2 Handling of Received DLLPs (Non-Flit Mode) §

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 3-20):

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

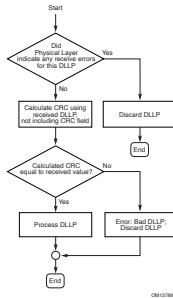


Figure 3-20 Received DLLP Error Check Flowchart §

- Received NOP DLLPs are discarded
 - Note: NOP2 DLLPs do not exist in Non-Flit Mode, as that encoding encoding is used for the Ack DLLP.
- 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 3-21):
 - If the $\uparrow\downarrow$ Sequence Number $\downarrow\uparrow$ $\uparrow\downarrow$ sequence number $\downarrow\uparrow$ specified by the AckNak_Seq_Num does not correspond to an unacknowledged TLP, or to the value in ACKD_SEQ , the DLLP is discarded
 - This is a Data Link Protocol Error , which is a reported error associated with the Port (see § Section 6.2).

Note that it is not an error to receive an Ack DLLP when there are no outstanding unacknowledged TLPs, including the time between reset and the first TLP transmission, as

long as the specified $\downarrow\downarrow$ Sequence Number $\downarrow\downarrow$ $\uparrow\uparrow$ sequence number $\uparrow\uparrow$ matches the value in ACKD_SEQ .

- If the AckNak_Seq_Num does not specify the $\downarrow\downarrow$ Sequence Number $\downarrow\downarrow$ $\uparrow\uparrow$ sequence number $\uparrow\uparrow$ of the most recently acknowledged TLP, then the DLLP acknowledges some TLPs in the retry buffer:
 - Purge from the retry buffer all TLPs from the oldest to the one corresponding to the AckNak_Seq_Num
 - Load ACKD_SEQ with the value in the AckNak_Seq_Num field
 - Reset REPLAY_NUM and REPLAY_TIMER
- If the DLLP is a Nak, initiate a replay (see above)

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

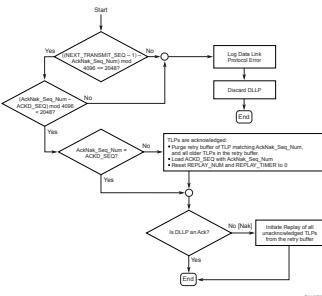


Figure 3-21 Ack/Nak DLLP Processing Flowchart §

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

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

3.6.2.3 Handling of Received DLLPs (Flit Mode) §

Since 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 Flow Control DLLPs received from the other component on the Link. These DLLPs are processed according to the following rules:

- In Flit Mode, detection of corrupt DLLPs occurs at the Flit level and there is no corruption check for DLLPs in the Data Link Layer.
- In Flit Mode, replay occurs at the Flit level and the Ack and Nak DLLPs are not used.
- DLLPs and Optimized_Update_FCs are not stored in the $\downarrow\downarrow$ Replay $\downarrow\downarrow$ $\uparrow\uparrow$ Retry $\uparrow\uparrow$ Buffer.
 - When a Flit is replayed, the DLP information is likely to be different from the original value.
 - Flit_Markers are associated with the TLP payload and are stored in the $\downarrow\downarrow$ Replay $\downarrow\downarrow$ $\uparrow\uparrow$ Retry $\uparrow\uparrow$ Buffer.
- A received DLLP that 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
- Received NOP DLLPs and NOP2 DLLPs are discarded

- Received FC DLLPs are passed to the Transaction Layer
- Received Optimized_Update_FCs are passed to the Transaction Layer
- Received PM DLLPs are passed to the component's power management control logic
- Received Link Management DLLPs are passed to the component's L0p control logic

3.6.3 LCRC and $\uparrow\downarrow TLP \uparrow$ Sequence Number (TLP Receiver) (Non-Flit Mode) §

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 $\uparrow\downarrow$ sequence number, $\uparrow\downarrow TLP$ Sequence Number, \uparrow passing the TLP to the Receive Transaction Layer if $\uparrow\downarrow OK \downarrow$ $\uparrow\downarrow ok \downarrow$ and requesting a replay if corrupted.

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

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

The following rules are applied in sequence to describe how received TLPs are processed, and what events trigger the transmission of Ack and Nak DLLPs (see § Figure 3-22):

- If the Physical Layer indicates a Receiver Error, discard any TLP currently being received and free any storage allocated for the TLP. Note that reporting such errors to software is done by the Physical Layer (and so are not reported by the Data Link Layer).
 - If a TLP was being received at the time the Receiver Error was indicated and the **NAK_SCHEDULED** flag is clear,
 - Schedule a Nak DLLP for transmission immediately
 - Set the **NAK_SCHEDULED** flag

- If the Physical Layer reports that the received TLP was nullified, and the LCRC is the logical NOT of the calculated value, discard the TLP and free any storage allocated for the TLP. This is not considered an error.
- If TLP was nullified but the LCRC does not match the logical NOT of the calculated value, the TLP is corrupt - discard the TLP and free any storage allocated for the TLP.

- If the NAK_SCHEDULED flag is clear,
 - Schedule a Nak DLLP for transmission immediately
 - Set the NAK_SCHEDULED flag

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

- The LCRC value is checked by:
 - Applying the same algorithm used for calculation (above) to the received TLP, not including the 32-bit LCRC field of the received TLP
 - Comparing the calculated result with the value in the LCRC field of the received TLP
 - if not equal, the TLP is corrupt - discard the TLP and free any storage allocated for the TLP
 - If the NAK_SCHEDULED flag is clear,
 - schedule a Nak DLLP for transmission immediately
 - set the NAK_SCHEDULED flag

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

- If the TLP Sequence Number is not equal to the expected value, stored in NEXT_RCV_SEQ:
 - Discard the TLP and free any storage allocated for the TLP
 - If the TLP Sequence Number satisfies the following equation:
 $(\text{NEXT_RCV_SEQ} - \text{TLP Sequence Number}) \bmod 4096 \leq 2048$
 the TLP is a duplicate, and an Ack DLLP is scheduled for transmission (per transmission priority rules)
 - Otherwise, the TLP is out of sequence (indicating one or more lost TLPs):
 - if the NAK_SCHEDULED flag is clear,
 - schedule a Nak DLLP for transmission immediately
 - set the NAK_SCHEDULED flag
 - This is a Bad TLP error and is a reported error associated with the Port (see § Section 6.2).
 - if the NAK_SCHEDULED flag is Set, the Port is permitted to, but is not recommended to, report a Bad TLP error associated with the Port (see § Section 6.2) and this permission is shown in § Figure 3-20.
- If the TLP Sequence Number is equal to the expected value stored in NEXT_RCV_SEQ:
 - The four bits, TLP Sequence Number, and LCRC (see § Figure 3-17) are removed and the remainder of the TLP is forwarded to the Receive Transaction Layer
 - The Data Link Layer indicates the start and end of the TLP to the Transaction Layer while transferring the TLP
 - The Data Link Layer treats the TLP as a “black box” and does not process or modify the contents of the TLP
 - Note that the Receiver Flow Control mechanisms do not account for any received TLPs until the TLP(s) are forwarded to the Receive Transaction Layer
 - NEXT_RCV_SEQ is incremented
 - If Set, the NAK_SCHEDULED flag is cleared

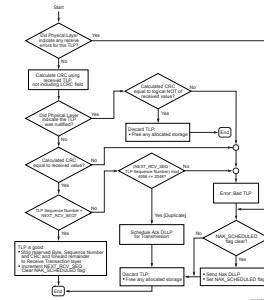


Figure 3-22 Receive Data Link Layer Handling of TLPs Flowchart §

- A TLP Receiver must schedule an Ack DLLP such that it will be transmitted no later than when all of the following conditions are true:
 - The Data Link Control and Management State Machine is in the DL_Active state
 - TLPs have been forwarded to the Receive Transaction Layer, but not yet acknowledged by sending an Ack DLLP
 - The AckNak_LATENCY_TIMER reaches or exceeds the value specified in § Table 3-10 for 2.5 GT/s operation, § Table 3-11 for 5.0 GT/s operation, § Table 3-12 for 8.0 GT/s and higher operation
 - The Link used for Ack DLLP transmission is already in L0 or has transitioned to L0
 - Note: if not already in L0, the Link must transition to L0 in order to transmit the Ack DLLP
 - Another TLP or DLLP is not currently being transmitted on the Link used for Ack DLLP transmission
 - The NAK_SCHEDULED flag is clear
 - Note: The AckNak_LATENCY_TIMER must be restarted from 0 each time an Ack or Nak DLLP is scheduled for transmission
- Data Link Layer Ack DLLPs may be scheduled for transmission more frequently than required
- Data Link Layer Ack and Nak DLLPs specify the value (NEXT_RCV_SEQ - 1) in the AckNak_Seq_Num field

§ Table 3-10 , § Table 3-11 , and § Table 3-12 define the threshold values for the AckNak_LATENCY_TIMER , which for any specific case is called the Ack Latency Limit.

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

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

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

For a Multi-Function Device where different Functions have different Rx_MPS_Limit values, it is strongly recommended that the smallest Rx_MPS_Limit value across all Functions be used.

Base 6.4 vs Base 6.3

*Table 3-10 Maximum Ack Latency Limits for 2.5 GT/s
(Symbol Times) (-0%/+0%) §*

| | | Link Operating Width | | | | |
|-------------------------|------|----------------------|------|------|-----|-----|
| | | x1 | x2 | x4 | x8 | x16 |
| Rx_MPS_Limit (bytes) | 128 | 237 | 128 | 73 | 67 | 48 |
| | 256 | 416 | 217 | 118 | 107 | 72 |
| | 512 | 559 | 289 | 154 | 86 | 86 |
| | 1024 | 1071 | 545 | 282 | 150 | 150 |
| | 2048 | 2095 | 1057 | 538 | 278 | 278 |
| | 4096 | 4143 | 2081 | 1050 | 534 | 534 |

*Table 3-11 Maximum Ack Latency Limits for 5.0 GT/s
(Symbol Times) (-0%/+0%) §*

| | | Link Operating Width | | | | |
|-------------------------|------|----------------------|------|------|-----|-----|
| | | x1 | x2 | x4 | x8 | x16 |
| Rx_MPS_Limit (bytes) | 128 | 288 | 179 | 124 | 118 | 99 |
| | 256 | 467 | 268 | 169 | 158 | 123 |
| | 512 | 610 | 340 | 205 | 137 | 137 |
| | 1024 | 1122 | 596 | 333 | 201 | 201 |
| | 2048 | 2146 | 1108 | 589 | 329 | 329 |
| | 4096 | 4194 | 2132 | 1101 | 585 | 585 |

*Table 3-12 Maximum Ack Latency Limits for 8.0 GT/s
and higher data rates (Symbol Times) §*

| | | Link Operating Width | | | | |
|-------------------------|------|----------------------|------|------|-----|-----|
| | | x1 | x2 | x4 | x8 | x16 |
| Rx_MPS_Limit (bytes) | 128 | 333 | 224 | 169 | 163 | 144 |
| | 256 | 512 | 313 | 214 | 203 | 168 |
| | 512 | 655 | 385 | 250 | 182 | 182 |
| | 1024 | 1167 | 641 | 378 | 246 | 246 |
| | 2048 | 2191 | 1153 | 634 | 374 | 374 |
| | 4096 | 4239 | 2177 | 1146 | 630 | 630 |

IMPLEMENTATION NOTE: RETRY BUFFER SIZING §

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

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

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

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

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

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

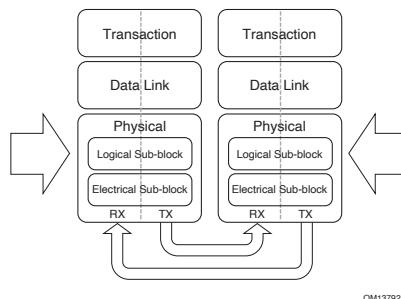
Retimers add latency (see § Section 4.3.9) and operating in SRIS can add latency. Implementations are strongly encouraged to consider these effects when determining the optimal buffer size.

Base 6.4 vs Base 6.3

4. Physical Layer Logical Block §

4.1 Introduction §

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



OM13792A

Figure 4-1 Layering Diagram Highlighting Physical Layer §

§ Chapter 4. describes the logical sub-block and § Chapter 8. describes the electrical sub-block.⁶⁵

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.

PCI Express uses three types of encoding (8b/10b, 128b/130b, and 1b/1b) and two Data Stream modes (Flit Mode and Non-Flit Mode). A Data Stream in the Non-Flit Mode is defined as a contiguous collection of TLPs, DLLPs, and Logical Idle/IDL Token, starting at the end of an Ordered Set and ending with another Ordered Set or a Link Electrical Idle. A Data Stream in Flit Mode is defined as a set of Flits (see § Section 4.2.3), starting at the end of the first SKP Ordered Set after an SDS Ordered Set , and ending with the last Flit prior to an Ordered Set other than SKP Ordered Set that causes the Link to exit out of L0 state or if the Link enters Electrical Idle. The encoding is determined by the Data Rate of the Link. The Data Stream mode is determined during initial Link training. If not disabled (see Flit Mode Disable) and if both the Ports (and all Pseudo-Ports, if any) support it, (see Flit Mode Supported), Flit Mode is chosen. Otherwise, Non-Flit Mode is chosen. See § Table 4-1 for valid encoding and symbol placement mode combinations.

65. Prior to [PCIe-4.0] § Chapter 4. described both logical and electrical sub-blocks. With [PCIe-4.0], the electrical section was moved to a new § Chapter 8. and a new Retimer section was added.

Table 4-1 Valid Encoding and Data Stream Mode Combinations §

| Current Data Rate | Flit Mode Negotiated during Configuration when LinkUp=0b | Encoding | Data Stream |
|--------------------------------|--|-----------|---------------|
| 2.5 GT/s, 5.0 GT/s | No | 8b/10b | Non-Flit Mode |
| 2.5 GT/s, 5.0 GT/s | Yes | 8b/10b | Flit Mode |
| 8.0 GT/s, 16.0 GT/s, 32.0 GT/s | No | 128b/130b | Non-Flit Mode |
| 8.0 GT/s, 16.0 GT/s, 32.0 GT/s | Yes | 128b/130b | Flit Mode |
| 64.0 GT/s | Yes (mandatory) | 1b/1b | Flit Mode |

The Ordered Set encoding follows the 8b/10b, 128b/130b, and 1b/1b encoding as defined in § Table 4-2 .

Table 4-2 Valid Encoding for Ordered Sets §

| Current Data Rate | Flit Mode Negotiated during Configuration when LinkUp=0b | Encoding | Comments |
|--------------------------------|--|-----------|---|
| 2.5 GT/s, 5.0 GT/s | Yes / No | 8b/10b | Same Ordered Sets are used in Flit Mode as well as Non-Flit Mode. |
| 8.0 GT/s | No | 128b/130b | Only Standard SKP Ordered Set sent when SKP OS needs to be sent. Rest of the Ordered Sets are identical for Non-Flit Mode and Flit Mode in 8.0 GT/s. |
| 16.0 GT/s, 32.0 GT/s | No | 128b/130b | Alternates between Standard SKP OS and Control SKP OS, when SKP OS needs to be sent. Rest of the Ordered Sets are identical for Non-Flit Mode and Flit Mode in the corresponding Data Rate. |
| 8.0 GT/s, 16.0 GT/s, 32.0 GT/s | Yes | 128b/130b | Alternates between Standard SKP OS and Control SKP OS, when SKP OS needs to be sent. Rest of the Ordered Sets are identical for Non-Flit Mode and Flit Mode in the corresponding Data Rate. |
| 64.0 GT/s | Yes (mandatory) | 1b/1b | All Ordered Sets follow 1b/1b encoding at 64.0 GT/s with PAM4 signaling. Only Control SKP OS sent when SKP OS needs to be sent. |

IMPLEMENTATION NOTE: FLIT MODE IDENTIFICATION THROUGHOUT THE DOCUMENT §

Support for Flit Mode behavior is referenced five times in the specification through the use of the following fields/variables:

Flit Mode Supported bit

Bit 0 of the Data Rate Identifier Symbol of 8b/10b and 128b/130b encoded TS1s and TS2s . This bit is Set when the Flit Mode Supported bit in the PCI Express Capabilities Register is Set and the Flit Mode Disable bit in the Link Control Register is Clear. See § Table 4-36 , § Table 4-37 , and § Table 4-38 .

Flit_Mode_Enabled

Variable that indicates whether or not Flit Mode has been successfully negotiated. See § Section 4.2.7.1.1 and § Section 4.2.7.3.2 .

Flit Mode Supported

Field in the PCI Express Capabilities Register . Flit Mode is supported when this bit is Set. See § Section 7.5.3.2 .

Flit Mode Disable

Field in the Link Control Register . This bit used to disable Flit Mode. This bit is most useful in Downstream Ports but is also defined for Upstream Ports (useful for crosslinks or by device firmware). See § Section 7.5.3.7 . Setting this bit may be useful to workaround faulty hardware.

Flit Mode Status

Field in the Link Status 2 Register . Indicates that ↑↓that↓ the Link is or will be operating in Flit Mode. Should match the Flit_Mode_Enabled variable. See § Section 7.5.3.20 .

4.2.1 8b/10b Encoding for 2.5 GT/s and 5.0 GT/s Data Rates §

4.2.1.1 Symbol Encoding §

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

Base 6.4 vs Base 6.3

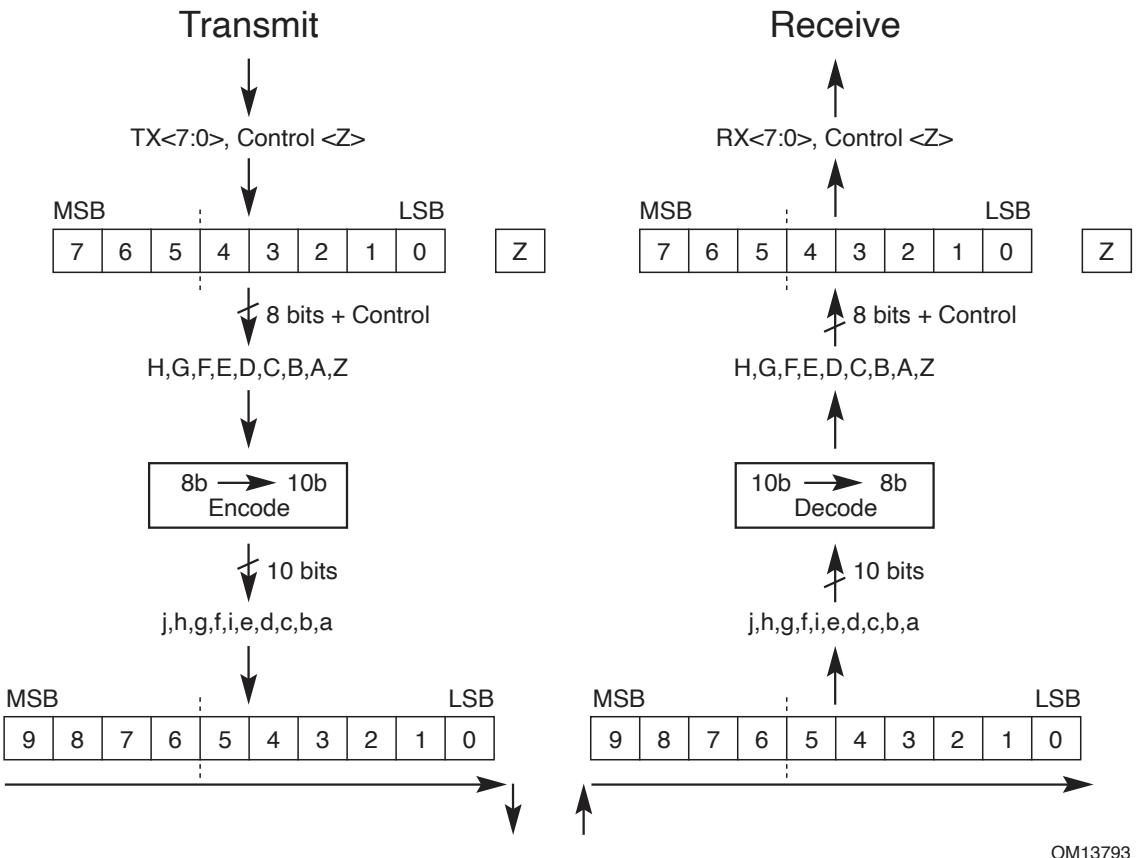


Figure 4-2 Character to Symbol Mapping §

4.2.1.1.1 Serialization and De-serialization of Data §

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

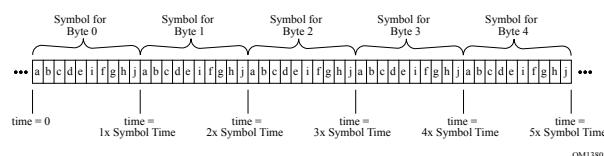
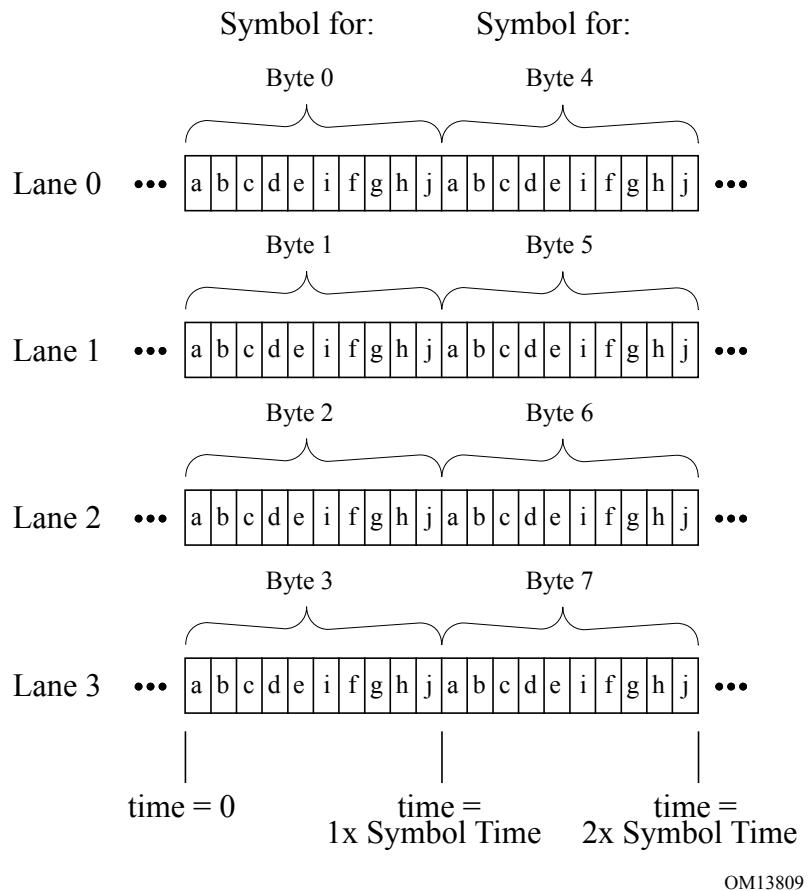


Figure 4-3 Bit Transmission Order on Physical Lanes - x1 Example §

Base 6.4 vs Base 6.3



OM13809

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

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

The 8b/10b encoding scheme provides Special Symbols that are distinct from the Data Symbols used to represent Characters. These Special Symbols are used for various Link Management mechanisms described later in this chapter. Special Symbols are also used to frame DLLPs and TLPs⁶⁶ in Non-Flit Mode, using distinct Special Symbols to allow these two types of Packets to be quickly and easily distinguished. When Flit Mode is enabled, each Symbol (Byte) of the Data Stream is still encoded with 8b/10b encoding without the Framing described. The Flit Mode operation is described in § Section 4.2.3.1. Even when Flit Mode is enabled, the Ordered Sets follow the description provided in § Section 4.2.1, when operating in 2.5 GT/s or 5.0 GT/s Data Rates.

§ Table 4-3 shows the Special Symbols used for PCI Express and provides a brief description for each. These Symbols will be discussed in greater detail in following sections. Each of these Special Symbols, as well as the data Symbols, must be interpreted by looking at the 10-bit Symbol in its entirety.

Table 4-3 Special Symbols in 8b/10b Encoding §

| Encoding | Symbol | Name | Description |
|----------|--------|-------|--|
| K28.5 | COM | Comma | Used for Lane and Link initialization and management |

66. In § Chapter 4., PMUX packets follow the TLP framing rules for Non-Flit Mode. PMUX is not supported in Flit Mode.

Base 6.4 vs Base 6.3

| Encoding | Symbol | Name | Description |
|----------|------------|-------------------------------|--|
| | | | Used identically in Flit Mode and Non-Flit Mode |
| K27.7 | STP | Start TLP | Marks the start of a Transaction Layer Packet in Non-Flit Mode. Reserved in Flit Mode. |
| K28.2 | SDP | Start DLLP | Marks the start of a Data Link Layer Packet in Non-Flit Mode. Reserved in Flit Mode. |
| K29.7 | END | End | Marks the end of a Transaction Layer Packet or a Data Link Layer Packet in Non-Flit Mode. Reserved in Flit Mode. |
| K30.7 | EDB | EnD Bad | Marks the end of a nullified TLP in Non-Flit Mode. Reserved in Flit Mode. |
| K23.7 | PAD | Pad | Used in Framing in Non-Flit Mode only. It is also used in Link Width and Lane ordering negotiations in both Non-Flit Mode and Flit Mode. |
| K28.0 | SKP | Skip | Used for compensating for different bit rates for two communicating Ports. Used identically in Flit Mode and Non-Flit Mode. |
| K28.1 | <u>FTS</u> | <u>Fast Training Sequence</u> | Used within an Ordered Set to exit from L0s to L0 in Non-Flit Mode. No usage for this Encoding is defined in Flit Mode. |
| K28.3 | IDL | Idle | Used in the <u>Electrical Idle Ordered Set (EIOS)</u> Used identically in Flit Mode and Non-Flit Mode. |
| K28.4 | | | Reserved |
| K28.6 | | | Reserved |
| K28.7 | EIE | Electrical Idle Exit | Reserved in 2.5 GT/s Used in the <u>Electrical Idle Exit Ordered Set (EIEOS)</u> and sent prior to sending <u>FTS</u> at data rates other than 2.5 GT/s |

4.2.1.1.3 8b/10b Decode Rules §

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

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

The initial disparity for a Receiver that detects an exit from Electrical Idle is set to the disparity of the first Symbol used to obtain Symbol lock. Disparity may also be re-initialized if Symbol lock is lost and regained during the transmission of differential information due to an implementation specific number of errors. All following received Symbols after the initial disparity is set must be found in the proper column corresponding to the current running disparity.

If a received Symbol is found in the column corresponding to the incorrect running disparity or if the Symbol does not correspond to either column, the Physical Layer must notify the Data Link Layer that the received Symbol is invalid. This is a Receiver Error, and is a reported error associated with the Port (see § Section 6.2) in Non-Flit Mode. In Flit Mode, the Symbol in error is passed to the FEC logic to correct; the Receiver is permitted to send any 8-bit value to the FEC logic if an 8b/10b error or k-char is detected inside the Flit boundary.

4.2.1.2 Framing and Application of Symbols to Lanes §

There are two classes of framing and application of Symbols to Lanes. The first class consists of the Ordered Sets. The second class consists of TLPs and DLLPs in the Data Stream. Ordered Sets are always transmitted serially on each Lane, such that a full Ordered Set appears simultaneously on all Lanes of a multi-Lane Link. The Non-Flit Mode of Data Stream is described below. The Flit Mode description for Data Stream is described in § Section 4.2.3.2 and § Section 4.2.3.3. There are no defined framing-related errors while using 8b/10b encoding in Flit Mode.

4.2.1.2.1 Framing and Application of Symbols to Lanes for TLPs and DLLPs in Non-Flit Mode §

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.

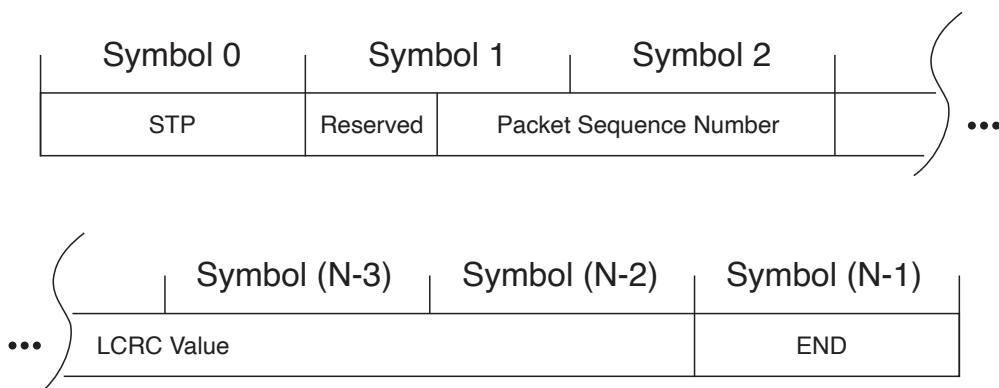
The conceptual stream of Symbols must be mapped from its internal representation, which is implementation dependent, onto the external Lanes. The Symbols are mapped onto the Lanes such that the first Symbol (representing Character 0) is placed onto Lane 0; the second is placed onto Lane 1; etc. The x1 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 (00h), scrambled according to the rules of § Section 4.2.1.3 and 8b/10b encoded according to the rules of § Section 4.2.1.1, in the same way that TLP and DLLP Data Symbols are scrambled and encoded. Likewise, when the Receiver is not receiving any packet information or special Ordered Sets, the Receiver is in Logical Idle and shall receive idle data as described above. During transmission of the idle data, the SKP Ordered Set must continue to be transmitted as specified in § Section 4.2.8.

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

- TLPs must be framed by placing an STP Symbol at the start of the TLP and an END Symbol or EDB Symbol at the end of the TLP (see § Figure 4-5).
- A properly formed TLP contains a minimum of 18 symbols between the STP and END or EDB Symbols. If a received sequence has less than 18 symbols between the STP and END or EDB symbols, the Receiver is permitted to treat this as a Receiver Error.
 - If checked, this is a reported error associated with the Receiving Port (see § Section 6.2).
- DLLPs must be framed by placing an SDP Symbol at the start of the DLLP and an END Symbol at the end of the DLLP (see § Figure 4-6).

- Logical Idle is defined to be a period of one or more Symbol Times when no information: TLPs, DLLPs or any type of Special Symbol is being Transmitted/Received. Unlike Electrical Idle, during Logical Idle the Idle data Symbol (00h) is being transmitted and received.
 - When the Transmitter is in Logical Idle, the Idle data Symbol (00h) shall be transmitted on all Lanes. This is scrambled according to the rules in § [Section 4.2.1.3](#).
 - Receivers must ignore incoming Idle data Symbols, and must not have any dependency other than scramble sequencing on any specific data patterns.
- For Links wider than x1, the STP Symbol (representing the start of a TLP) must be placed in Lane 0 when starting Transmission of a TLP from a Logical Idle Link condition.
- For Links wider than x1, the SDP Symbol (representing the start of a DLLP) must be placed in Lane 0 when starting Transmission of a DLLP from a Logical Idle Link condition.
- The STP Symbol must not be placed on the Link more frequently than once per Symbol Time.
- The SDP Symbol must not be placed on the Link more frequently than once per Symbol Time.
- As long as the above rules are satisfied, TLP and DLLP Transmissions are permitted to follow each other successively.
- One STP Symbol and one SDP Symbol may be placed on the Link in the same Symbol Time.
 - Links wider than x4 can have STP and SDP Symbols placed in Lane 4^*N , where N is a positive integer. For example, for x8, STP and SDP Symbols can be placed in Lanes 0 and 4; and for x16, STP and SDP Symbols can be placed in Lanes 0, 4, 8, or 12.
- For xN Links where N is 8 or more, if an END or EDB Symbol is placed in a Lane K, where K does not equal N-1, and is not followed by an STP or SDP Symbol in Lane K+1 (i.e., there is no TLP or DLLP immediately following), then PAD Symbols must be placed in Lanes K+1 to Lane N-1.
 - For example, on a x8 Link, if END or EDB is placed in Lane 3, PAD must be placed in Lanes 4 to 7, when not followed by STP or SDP.
- The EDB Symbol is used to mark the end of a nullified TLP. Refer to [§ Section 3.6.2.1](#) for information on the usage of EDB.
- Receivers may optionally check for violations of the rules of this section. These checks are independently optional (see § [Section 6.2.3.4](#)). If checked, violations are Receiver Errors, and are reported errors associated with the Port (see § [Section 6.2](#)).



OM13794

Figure 4-5 TLP with Framing Symbols Applied

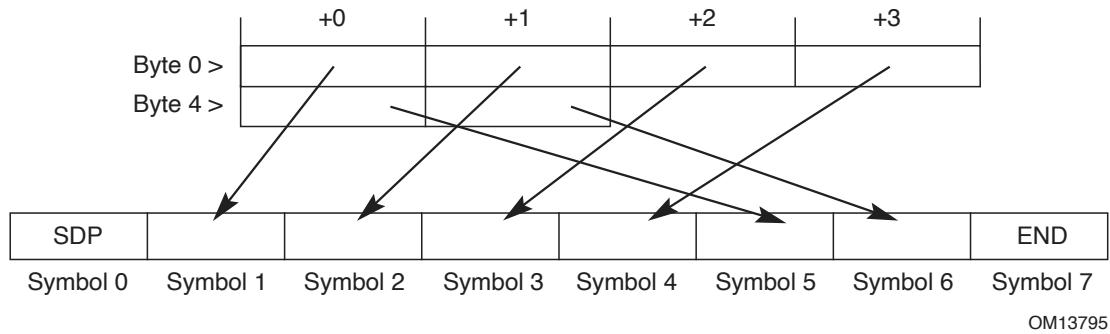


Figure 4-6 DLLP with Framing Symbols Applied §

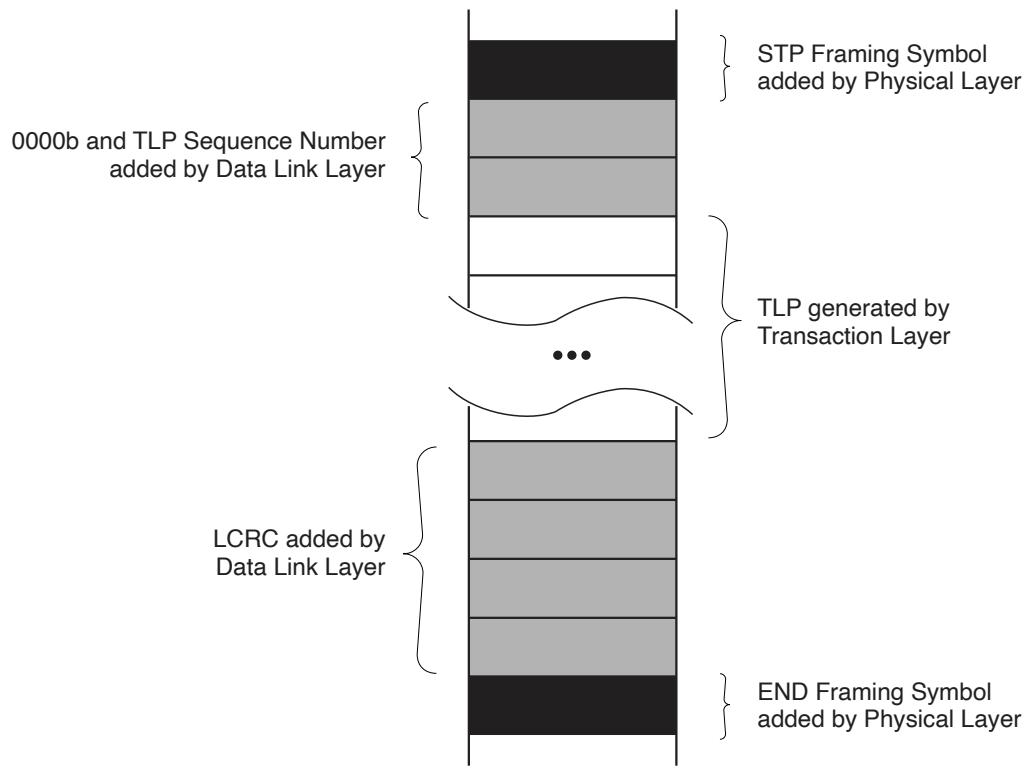
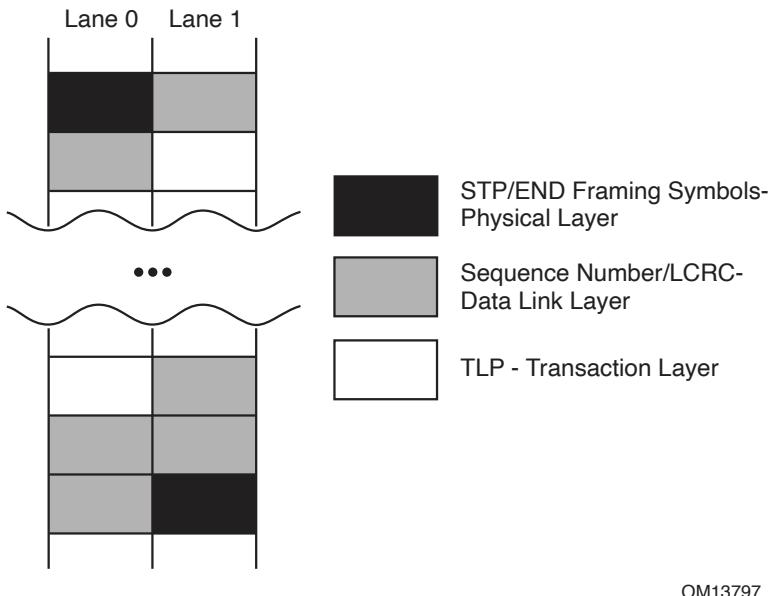


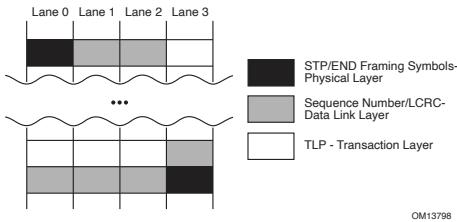
Figure 4-7 Framed TLP on a x1 Link §

Base 6.4 vs Base 6.3



OM13797

Figure 4-8 Framed TLP on a x2 Link §



OM13798

Figure 4-9 Framed TLP on a x4 Link §

4.2.1.3 Data Scrambling §

In order to improve electrical characteristics of a Link, data is typically scrambled. This is applicable for both the Flit Mode as well as Non-Flit Mode at 2.5 GT/s and 5.0 GT/s Data Rates. This involves XORing the data stream with a pattern generated by a Linear Feedback Shift Register (LFSR). On the Transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the Receive side, de-scrambling is applied to characters after 8b/10b decoding.

On a multi-Lane Link, the scrambling function can be implemented with one or many LFSRs. When there is more than one Transmit LFSR per Link, these must operate in concert, maintaining the same simultaneous (Lane-to-Lane Output Skew) value in each LFSR. When there is more than one Receive LFSR per Link, these must operate in concert, maintaining the same simultaneous (Lane-to-Lane Skew) value in each LFSR. Regardless of how they are implemented, LFSRs must interact with data on a Lane-by-Lane basis as if there was a separate LFSR as described here for each Lane within that Link.

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

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

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

The data scrambling rules are the following:

- The COM Symbol initializes the LFSR.
- The LFSR value is advanced eight serial shifts for each Symbol except the SKP.
- All data Symbols (D codes) except those within Ordered Sets (e.g., TS1, TS2, EIEOS), the Compliance Pattern (see § [Section 4.2.9](#)), and the Modified Compliance Pattern (see § [Section 4.2.10](#)) are scrambled.
- All special Symbols (K codes) are not scrambled.
- The initialized value of an LFSR seed (D0-D15) is FFFFh. Immediately after a COM exits the Transmit LFSR, the LFSR on the Transmit side is initialized. Every time a COM enters the Receive LFSR on any Lane of that Link, the LFSR on the Receive side is initialized.
- Scrambling can only be disabled at the end of Configuration (see § [Section 4.2.7.3.5](#)).
- Scrambling does not apply to a [Loopback Follower](#).
- Scrambling is always enabled in Detect by default.

IMPLEMENTATION NOTE: DISABLING SCRAMBLING §

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

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

For more information on scrambling, see § [Appendix C](#).

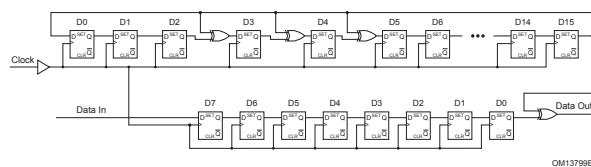


Figure 4-10 LFSR with 8b/10b Scrambling Polynomial §

4.2.2 128b/130b Encoding for 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s Data Rates §

When a PCI Express Link is operating at a data rate of 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s, it uses the 128b/130b encoding rules described in this section. For backwards compatibility, the Link initially trains to L0 at the 2.5 GT/s data rate using 8b/10b encoding as described in § [Section 4.2.1](#), then when the data rate is changed to 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s, 128b/130b encoding is used. 128b/130b encoding uses a Link-wide packetization mechanism in Non-Flit Mode and a

per-Lane block code with scrambling in both Flit Mode and Non-Flit Modes. In the Flit Mode, the Data Stream follows the same mechanism described in § Section 4.2.3.1 , for the 128b payload inside the 128b/130b Data Block(s).

The basic entity of data transmission is an 8-bit data character, referred to as a Symbol, as shown in § Figure 4-11 and § Figure 4-12 .

IMPLEMENTATION NOTE: SYMBOL IN 128B/130B ENCODING SCHEME §

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

4.2.2.1 Lane Level Encoding §

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

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

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

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

The bit transmission order is as follows. A Sync Header represented as ‘H₁ H₀’ is placed on a Lane starting with ‘H₀’ and ending with ‘H₁’. A Symbol, represented as ‘S₇ S₆ S₅ S₄ S₃ S₂ S₁ S₀’, is placed on a Lane starting with ‘S₀’ and ending with ‘S₇’. In the diagrams that show a time scale, bits represent the transmission order. In layout diagrams, bits are arranged in little-endian format, consistent with packet layout diagrams in other chapters of this specification.

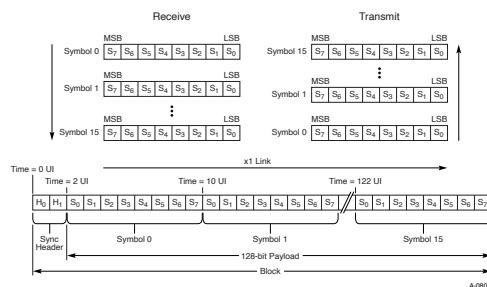


Figure 4-11 Example of Bit Transmission Order in a x1 Link Showing 130 Bits of a Block §

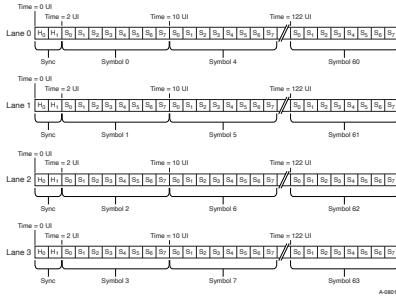


Figure 4-12 Example of Bit Placement in a x4 Link with One Block per Lane §

4.2.2.2 Ordered Set Blocks §

An Ordered Set Block contains a Sync Header followed by one Ordered Set. All Lanes of a multi-Lane Link must transmit the same Ordered Set type simultaneously. The first Symbol of the Ordered Set defines the type of Ordered Set. Subsequent symbols of the Ordered Set are defined by the Ordered Set type and need not be identical across lanes of a multi-Lane Link. The Ordered Sets are described in detail in § Section 4.2.5 and § Section 4.2.8 . Ordered Set Blocks are the same for both Flit Mode and Non-Flit Mode except for the use and frequency of SKP Ordered Set . In Flit Mode at 8.0 GT/s, both Standard SKP Ordered Sets and Control SKP Ordered Sets are used.

4.2.2.2.1 Block Alignment §

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

Unaligned Phase

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

Aligned Phase

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

Locked Phase

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

IMPLEMENTATION NOTE: DETECTION OF LOSS OF BLOCK ALIGNMENT §

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

Additional Requirements:

- While in the Aligned or Locked phase, Receivers must adjust their alignment as necessary when a SKP Ordered Set is received. See § Section 4.2.8 for more information on SKP Ordered Sets .
- After any LTSSM transition to Recovery, Receivers must ignore all received TS Ordered Sets until they receive an EIEOS . Conceptually, receiving an EIEOS validates the Receiver's alignment and allows TS Ordered Set processing to proceed. If a received EIEOS initiates an LTSSM transition from L0 to Recovery, Receivers are permitted to process any TS Ordered Sets that follow the EIEOS or ignore them until another EIEOS is received after entering Recovery.
- Receivers are permitted to transition from the Locked phase to the Unaligned or Aligned phase as long as Data Stream processing is stopped. See § Section 4.2.2.3 for more information on Data Stream requirements.
- Loopback Leads : While in Loopback.Entry , Leads must be capable of adjusting their Receiver's Block alignment to received EIEOS bit patterns. While in Loopback.Active, Leads are permitted to transmit an EIEOS and adjust their Receiver's Block alignment to the looped back bit stream.
- Loopback Followers : While in Loopback.Entry , Followers must be capable of adjusting their Receiver's Block alignment to received EIEOS bit patterns. While in Loopback.Active, Followers must not adjust their Receiver's Block alignment. Conceptually, the Receiver is directed to the Locked phase when the Follower starts to loop back the received bit stream.

4.2.2.3 Data Blocks §

The payload of Data Blocks is a stream of Symbols defined as a “Data Stream”. In Non-Flit Mode, the Data Stream consists of Framing Tokens, TLPs, and DLLPs. In Flit Mode, the Data Stream is described in § Section 4.2.3.1 . Each Symbol of the Data Stream is placed on a single Lane of the Link, and the stream of Symbols is striped across all Lanes of the Link and spans Block boundaries.

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

4.2.2.3.1 Framing Tokens in Non-Flit Mode ↑↑Non-Flit Mode↑↑Non-Flit Mode↑ §

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

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

Table 4-4 Framing Token Encoding §

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

Base 6.4 vs Base 6.3

| | | | | | | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| | +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 | 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 | 7 6 5 4 3 2 1 0 |
| TLP Length[3:0] | 1111b | F _P | TLP Length[10:4] | FCRC | | | TLP Sequence Number | |

STP Token

| | | | | | | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| | +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 | 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 | 7 6 5 4 3 2 1 0 |
| 0001b | 1111b | 1b | 0000000b | 1001b | | | 00000000000000b | |

EDS Token

| | | | | | | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| | +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 | 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 | 7 6 5 4 3 2 1 0 |
| 11000000b | | 11000000b | | 11000000b | | 11000000b | | |

EDB Token

| | | | | |
|-------------------------------|-------------------------------|-----------|----|--|
| | +0 | | +1 | |
| 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | | | |
| 11110000b | | 10101100b | | |

SDP Token

| | | |
|-------------------------------|----|--|
| | +0 | |
| 7 6 5 4 3 2 1 0 | | |
| 00000000b | | |

IDL Token

A-0802

Figure 4-13 Layout of Framing Tokens §

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

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

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

$$C[0] = L[10] \wedge L[7] \wedge L[6] \wedge L[4] \wedge L[2] \wedge L[1] \wedge L[0]$$

$$C[1] = L[10] \wedge L[9] \wedge L[7] \wedge L[5] \wedge L[4] \wedge L[3] \wedge L[2]$$

$$C[2] = L[9] \wedge L[8] \wedge L[6] \wedge L[4] \wedge L[3] \wedge L[2] \wedge L[1]$$

$$C[3] = L[8] \wedge L[7] \wedge L[5] \wedge L[3] \wedge L[2] \wedge L[1] \wedge L[0]$$

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

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

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

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

IMPLEMENTATION NOTE: FRAME CRC AND FRAME PARITY §

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

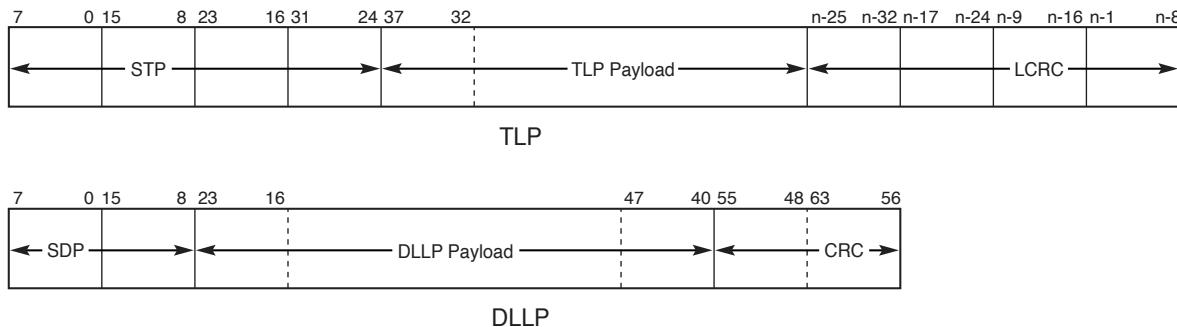
Note that, for TLPs, the Data Link Layer prepends 4 Reserved bits (0000b) to the TLP Sequence Number field before it calculates the LCRC. These Reserved bits are not explicitly transmitted when using 128b/130b encoding, and Receivers assume that the 4 bits received are 0000b when calculating the LCRC.

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

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

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

§ Figure 4-17 shows an example of TLPs, DLLPs, IDLs, and an EDS Token followed by a SKP Ordered Set. SKP Ordered Sets are defined in § Section 4.2.8.2.



A-0803

Figure 4-14 TLP and DLLP Layout §

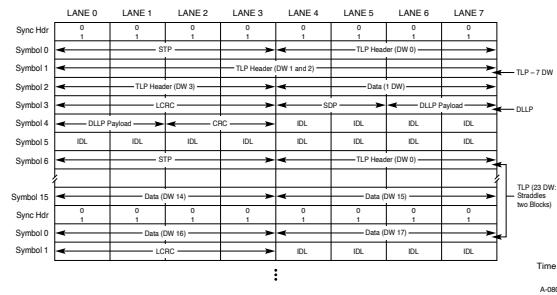


Figure 4-15 Packet Transmission in a x8 Link §

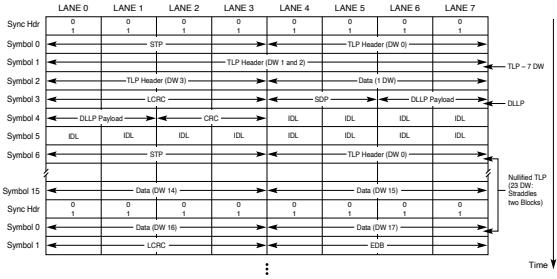


Figure 4-16 Nullified TLP Layout in a x8 Link with Other Packets §

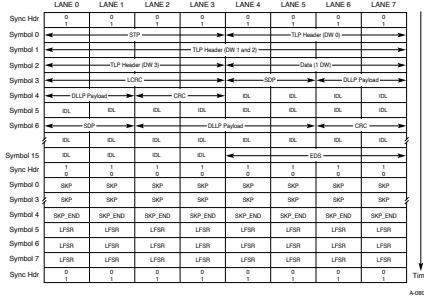


Figure 4-17 SKP Ordered Set of Length 66-bit in a x8 Link §

4.2.2.3.2 Transmitter Framing Requirements in Non-Flit Mode §

The following requirements apply to the transmitted Data Stream.

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

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

4.2.2.3.3 Receiver Framing Requirements in Non-Flit Mode §

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

- When processing Symbols that are expected to be a Framing Token, receiving a Symbol or sequence of Symbols that does not match the definition of a Framing Token is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error in the Lane Error Status Register for the Lane that receives the first Symbol of an expected Framing Token when that Symbol does not match Symbol +0 of an STP (bits [3:0] only), IDL, SDP, EDB, or EDS Token (see § Figure 4-13).
- All optional error checks and error reports in this section are independently optional (see § Section 6.2.3.4).
- When an STP Token is received:
 - Receivers must calculate the Frame CRC and Frame Parity of the received TLP Length field and compare the results to the received Frame CRC and Frame Parity fields. A Frame CRC or Frame Parity mismatch is a Framing Error.
 - An STP Token with Framing Error is not considered part of a TLP for the purpose of reporting to the Data Link Layer.
 - If the TLP Length field is 1, the Symbols are not an STP Token and are instead evaluated to determine whether they are an EDS Token.
 - Receivers are permitted to check whether the TLP Length field has a value of 0. If checked, receiving a TLP Length field of 0 is a Framing Error.
 - Receivers are permitted to check whether the TLP Length field has a value of 2, 3, or 4. If checked, receiving such a TLP Length field is a Framing Error.
 - Receivers are permitted to check whether the TLP Length field has a value between 1152 and 1535 (inclusive). If checked, receiving such a TLP Length field is a Framing Error.
 - Receivers on Ports that do not support Protocol Multiplexing are permitted to check whether the TLP Length field has a value greater than 1535. If checked, receiving such a TLP Length field is a Framing Error.
 - Receivers on Ports that support Protocol Multiplexing, shall process STP Tokens with a TLP Length field that is greater than 1535 as the start of a PMUX Packet as defined in § Appendix G..
 - The next Token to be processed begins with the Symbol immediately following the last DW of the TLP, as determined by the TLP Length field.

- Base 6.4 vs Base 6.3**
- Receivers must evaluate this Symbol and determine whether it is the first Symbol of an EDB Token and therefore whether the TLP is nullified. See the EDB Token requirements.
 - Receivers are permitted to check whether more than one STP Token is received in a single Symbol Time. If checked, receiving more than one STP Token in a single Symbol Time is a Framing Error.
 - When an EDB Token is received:
 - If an EDB Token is received immediately following a TLP (there are no Symbols between the last Symbol of the TLP and the first Symbol of the EDB Token), receivers must inform the Data Link Layer that an EDB Token has been received. Receivers are permitted to inform the Data Link Layer that an EDB Token has been received after processing the first Symbol of the EDB Token or after processing any or all of the remaining Symbols of the EDB Token. Regardless of when they inform the Data Link Layer of a received EDB Token, Receivers must check all Symbols of the EDB Token. Receiving a Symbol that does not match the definition of an EDB Token is a Framing Error.
 - Receiving an EDB Token at any time other than immediately following a TLP is a Framing Error.
 - The next Token to be processed begins with the Symbol immediately following the EDB Token.
 - When an EDS Token is received in the last four Symbols of the Data Block across the Link:
 - Receivers must stop processing the Data Stream.
 - Receiving an Ordered Set other than SKP, EIOS, or EIEOS in the Block following the EDS Token is a Framing Error.
 - If a SKP Ordered Set is received in the Block following the EDS Token, Receivers resume Data Stream processing with the first Symbol of the Data Block that follows the SKP Ordered Set unless a Framing Error has been detected.
 - When an SDP Token is received:
 - The next Token to be processed begins with the Symbol immediately following the last Symbol of the DLLP.
 - Receivers are permitted to check whether more than one SDP Token is received in a single Symbol Time. If checked, receiving more than one SDP Token in a single Symbol Time is a Framing Error.
 - When an IDL Token is received:
 - For a x1 Link, the next Token to be processed begins with the next Symbol received.
 - For a x2 Link, the next Token to be processed begins with the Symbol received in Lane 0 of the next Symbol Time. It is strongly recommended that Receivers check whether the Symbol received in Lane 1, if it did not receive IDL, after an IDL Token was received in Lane 0 is also IDL and report an error for Lane 1 in the Lane Error Status Register. If checked, receiving a Symbol other than IDL is a Framing Error.
 - For a x4 Link, the next Token to be processed begins with the Symbol received in Lane 0 of the next Symbol Time. It is strongly recommended that Receivers check whether the Symbols received in Lanes 1-3, after an IDL Token was received in Lane 0 are also IDL and report an error for the Lane(s) that did not receive IDL, in the Lane Error Status Register. If checked, receiving a Symbol other than IDL is a Framing Error.
 - For x8 and x16 Links, the next Token to be processed begins with the Symbol received in the next DW aligned Lane following the IDL Token. For example, if an IDL Token is received in Lane 4 of a x16 Link, the next Token location begins with Lane 8 of the same Symbol Time. However, if an IDL Token is received on Lane 4 of a x8 Link, the next Token location begins with Lane 0 of the following Symbol Time. It is strongly recommended that Receivers check whether the Symbols received between the IDL Token and the next Token location are also IDL and report an error for the Lane(s) that did not receive IDL, in the Lane Error Status Register. If checked, receiving a Symbol other than IDL is a Framing Error.

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

- While processing the Data Stream, Receivers must also check the Block type received by each Lane, after accounting for Lane-to-Lane de-skew, for the following conditions:
 - Receiving an Ordered Set Block on any Lane immediately following an SDS Ordered Set is a Framing Error.
 - Receiving a Block with an undefined Block type (a Sync Header of 11b or 00b) is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error for any Lane that received the undefined Block type in the Lane Error Status register.
 - Receiving an Ordered Set Block on any Lane without receiving an EDS Token in the preceding Block is a Framing Error. For example, receiving a SKP Ordered Set without a preceding EDS Token is a Framing Error. In addition, receiving a SKP Ordered Set followed immediately by another Ordered Set Block (including another SKP Ordered Set) within a Data Stream is a Framing Error. It is strongly recommended that if the first Symbol of the Ordered Set is SKP, Receivers of a multi-Lane Link report an error for the Lane(s) in the Lane Error Status register if the received Symbol number 1 through 4N does not match the corresponding Symbol in § Table 4-64 or § Table 4-65 .
 - Receiving a Data Block on any Lane when the previous block contained an EDS Token is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error for the Lane(s) that received the Data Block in the Lane Error Status register.
 - Receivers are permitted to check for different Ordered Sets on different Lanes. If checked, receiving different Ordered Sets is a Framing Error. For example, if Lane 0 receives a SKP Ordered Set and Lane 1 receives an EIOS , it would be a Framing Error.

4.2.2.3.4 Receiver Framing Requirements in Flit Mode §

While processing the Data Stream, Receivers must check the Block type received by each Lane, after accounting for Lane-to-Lane de-skew, for the following conditions:

- Not receiving a SKP Ordered Set followed by a Data Block on any Lane immediately following an SDS Ordered Set is a Framing Error.
- Receiving a Block with an undefined Block type (a Sync Header of 11b or 00b) is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error for any Lane that received the undefined Block type in the Lane Error Status register.
- For Lanes that are not entering or exiting the electrical idle state as part of L0p state, the following conditions result in a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error for any Lane that caused the Framing Error in the Lane Error Status register.
 - Receiving an Ordered Set Block on any Lane in an unscheduled Block boundary, as defined in § Section 4.2.3.1
 - Not receiving one of these three Ordered Sets of the appropriate length at the scheduled Block boundary: EIEOS , SKP Ordered Set , or EIOS
 - Receivers are permitted to check for different Ordered Sets on different Lanes, except as permitted for an L0p width transition. If checked, receiving different Ordered Sets is a Framing Error. For example, if Lane 0 receives a SKP Ordered Set and Lane 1 receives an EIOS and no L0p transition is expected, it would be a Framing Error.

4.2.2.3.5 Recovery from Framing Errors in Non-Flit Mode and Flit Mode §

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

- Report a Receiver Error as described in § Section 4.2.5.8 .
- Stop processing the Data Stream. Processing of a new Data Stream is initiated when the next SDS Ordered Set is received as previously described.
- Initiate the error recovery process as described in § Section 4.2.5.8 . If the LTSSM state is L0, direct the LTSSM to Recovery. If the LTSSM state is Configuration.Complete or Configuration.Idle when the Framing Error is detected, the error recovery process is satisfied by either a transition from Configuration.Idle to Recovery.RcvrLock due to the specified timeout, or a transition to Recovery through L0. If the LTSSM state is Recovery.RcvrCfg or Recovery.Idle when the Framing Error is detected, the error recovery process is satisfied by either a transition from Recovery.Idle to Recovery.RcvrLock due to the specified timeout, or a directed transition from L0 to Recovery. If the LTSSM substate is either Recovery.RcvrLock or Configuration.Linkwidth.Start, the error recovery process is satisfied upon exit from these substates and no direction of the LTSSM to Recovery is required.
 - Note: The framing error recovery mechanism is not expected to directly cause any Data Link Layer initiated recovery action such as NAK.

IMPLEMENTATION NOTE: TIME SPENT IN RECOVERY DUE TO DETECTION OF A FRAMING ERROR §

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

4.2.2.4 Scrambling in Non-Flit Mode and Flit Mode §

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

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

The scrambling rules are as follows:

- The two bits of the Sync Header used in 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s Data Rate are not scrambled and do not advance the LFSR.
- All 16 Symbols of an Electrical Idle Exit Ordered Set (EIEOS) bypass scrambling. Except during L0p , the scrambling LFSR is initialized after the last Symbol of an EIEOS is transmitted, and the descrambling LFSR is initialized after the last Symbol of an EIEOS is received.
- TS1 and TS2 Ordered Sets for 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s Data Rate:
 - Symbol 0 of a TS1 or TS2 Ordered Set bypasses scrambling.
 - Symbols 1-13 are scrambled.

- Symbols 14 and 15 bypass scrambling if required for DC Balance, but they are scrambled if not required for DC Balance.
- All 16 Symbols of a Fast Training Sequence (FTS) Ordered Set used in 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s bypass scrambling.
- All 16 Symbols of a Start of Data Stream (SDS) Ordered Set bypass scrambling.
- All 16 Symbols of an Electrical Idle Ordered Set (EIOS) bypass scrambling.
- All Symbols of a SKP Ordered Set bypass scrambling.
- Transmitters advance their LFSR for all Symbols of all Ordered Sets except for the SKP Ordered Set. The LFSR is not advanced for any Symbols of a SKP Ordered Set.
- Receivers evaluate Ordered Sets to determine whether to advance their LFSR. If the Ordered Set is a SKP Ordered Set (see § Section 4.2.8), or an EIOS with L0p in the midst of a data stream then the LFSR is not advanced for any Symbol of the Ordered Set. Otherwise, the LFSR is advanced for all Symbols of the Ordered Set.
- In L0p, the scrambler on the Transmit and Receive sides associated with all the Lanes must advance whenever the scrambler associated with Lane 0 advances and does not advance whenever the scrambler associated Lane 0 does not advance. As a result, in L0p, the transmit or receipt of an EIEOS does not reset the scrambler associated with the Lane, if it is not Lane 0.

IMPLEMENTATION NOTE: ELASTIC BUFFER AND L0P §

For implementations that descramble prior to putting entries into the elastic buffer, one way to handle the scramblers advancing in the idle lane during L0p is as follows:

- The Lane which goes to electrical idle will tie its decision to advance the scrambler, accounting for sync hdr, if any, and SKP Ordered Set to Lane 0. Since the EIOS in the Lane coincides with SKP OS in Lane 0, the scrambler in the idle Lane should be synchronized with the Lane 0 on receipt of EIOS
- The TS1 / TS2 Ordered Sets exchanged when trying to upsize the Link on L0p will be scrambled using the continually running LFSR.
- Depending on the implementation, any differences in scrambler synchronization on the receive side for lanes which have recently become active can be determined on the SKP Ordered Set boundary following block alignment and adjusted accordingly.
- The special scrambler rules for L0p above apply only to 128b/130b and 1b/1b encoding since the Ordered Sets are scrambled, and different Lanes can have different seeds/ taps which must run continuously without being reset during the data stream. In contrast, 8b/10b encoding does not scramble Ordered Sets and all the scramblers across all Lanes will be reset during SKP Ordered Sets during the link width up-size.

- All 16 Symbols of a Data Block are scrambled and advance the scrambler.
- For Symbols that need to be scrambled, the least significant bit is scrambled first and the most significant bit is scrambled last.
- The seed value of the LFSR is dependent on the Lane ↓↑number Default Lane numbers are assigned in an implementation specific manner which is invariant to Link width and Lane reversal negotiation. These default numbers must be unique. For example, each Lane↓ ↑number. The initial seed value↑ of ↓↑a x16 Link must be assigned a unique Lane number from 0 to

Errata: Base 6.3
B825△◀▷

15. These default Lane ^{↑↑the LFSR is determined using the Physical Lane Numbers . The LFSR seed↑}
numbers are reassigned ^{↑↑to} ^{↑↑using} ^{↑↑the Negotiated} ^{↑↑Lane} ^{↑↑Numbers} when the Link first enters Configuration.Idle (i.e., having gone through ^{↑↑Polling} ^{↑↑Polling} from ^{↑↑Detect} ^{↑↑Detect} with ^{↑↑LinkUp} ^{↑↑LinkUp} = 0b). ^{↑↑See also} **IMPLEMENTATION NOTE: LFSR IMPLEMENTATION WITH A SHARED LFSR.**

- The seed values for Lane number modulo 8 are:

| Lane | Seed |
|------|---------|
| 0 | 1DBFBCh |
| 1 | 0607BBh |
| 2 | 1EC760h |
| 3 | 18C0DBh |
| 4 | 010F12h |
| 5 | 19CFC9h |
| 6 | 0277CEh |
| 7 | 1BB807h |

IMPLEMENTATION NOTE: SCRAMBLING PSEUDO-CODE §

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

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

Base 6.4 vs Base 6.3

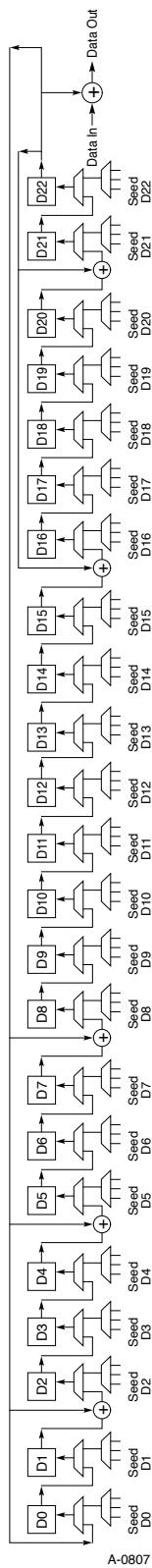
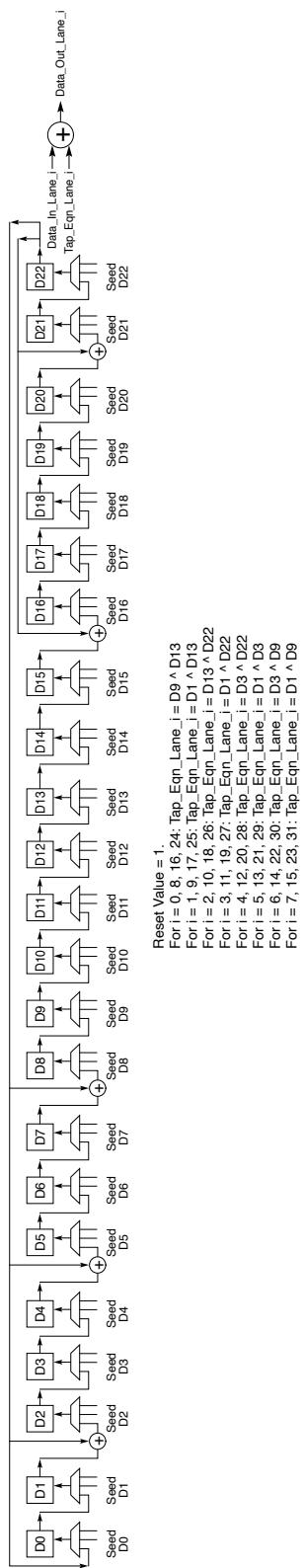


Figure 4-18 LFSR with Scrambling Polynomial in 8.0 GT/s and Above Data Rate §

IMPLEMENTATION NOTE: LFSR IMPLEMENTATION WITH A SHARED LFSR §

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

Base 6.4 vs Base 6.3



A-0808

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

4.2.2.5 Precoding §

A Receiver may request precoding from its transmitter for operating at data rates of 32.0 GT/s and higher. Precoding, when enabled at a Data Rate, applies to both Flit Mode and Non-Flit Mode at that data rate. The precoding rules are as follows:

- A Port or Pseudo-Port must request precoding on all configured Lanes of the Link. Behavior is undefined if precoding is requested on some Lanes but not others by a Port or Pseudo-Port.
- A Port or Pseudo-Port may request precoding independent of other Ports or Pseudo-Ports. For example, it is possible that precoding may be turned on only in the Upstream Port in the case with no Retimers in § Figure 4-79 , or on all the Lanes in Tx(A) and Tx(E) in the two Retimer example in § Figure 4-79 .
- Precoding is turned off for all data rates when the LTSSM is in the Detect state.
- If a precoding request for a data rate is to be made, it must be made prior to entering that data rate. A precoding request is made by setting the Transmitter Precode Request bit in the EQ TS2 or the 128b/130b EQ TS2 Ordered Sets prior to the transition to Recovery.Speed for the data rate at which the precoding will be turned on. ↑↓Prior to the first Link speed negotiation to 32.0 GT/s since exiting Detect, the Transmitter Precode Request bit represents the precoding request for 32.0 GT/s when the Supported Link Speeds field in the EQ TS2 or the 128b/130b EQ TS2 Ordered sets is 32.0 GT/s or above. If the Link speed has been negotiated to 32.0 GT/s since exiting Detect, the ↑↓The Transmitter Precode Request bit represents the precoding request for the same data rate that the values ↑↓in ↓↑to which↑ the Transmitter Preset field ↑↓apply to.↑ ↑↓apply.↑ For each data rate of 32.0 GT/s or higher, the precoding request must be made independently.
- Each (pseudo) Port must store the precoding request along with the Tx Eq values for each Lane from the most recent successful equalization procedure prior to the current transition through Detect . If the Link operates at 32.0 GT/s or higher data rate without performing equalization through the No Equalization Needed mechanism it negotiated in the TS1 / TS2 Ordered Sets or modified TS1/TS2 Ordered Sets or the Link operates at the 32.0 GT/s or higher data rate in Polling.Compliance or Loopback , the precoding requests from the stored equalization results must be enforced. If no equalization has ever been performed on the Link (prior to the current Link up), then precoding will not be turned on.
- If the Transmitter Precode Request bit is Set to 1b in each of the received eight consecutive EQ TS2 or 128b/130b EQ TS2 Ordered Sets during Recovery.RcvrCfg prior to entry to Recovery.Speed , the Transmitter must turn on the precoding for the data rate at which the Link will operate on exit from Recovery.Speed if the data rate is 32.0 GT/s or higher and the precoding request was intended for the data rate of operation. If the precoding request applies to a speed other than the data rate at which the Link will operate on exit from Recovery.Speed , the received Transmitter Precode Request bit must be ignored and the Precoding setting must not be updated for any data rate. Once turned on, precoding will be in effect for that data rate until the Transmitter receives another set of eight consecutive EQ TS2 or 128b/130b EQ TS2 Ordered Sets with Transmitter Precode Request set to 0b during Recovery.RcvrCfg prior to entry to Recovery.Speed for the same data rate.
- A Transmitter must not turn on precoding for any data rates lower than 32.0 GT/s.
- In data rates of 32.0 GT/s or higher, a Transmitter must set the Transmitter Precoding On bit to 1b in the TS1 Ordered Sets that it transmits in Recovery if its precoding is on for the current data rate; else the bit must be set to 0b.
- A Transmitter that has turned on precoding for the 32.0 GT/s data rate on Lane 0 must set the 32.0 GT/s Transmitter Precoding On bit to 1b in the 32.0 GT/s Status Register; else, it must set the bit to 0b. A Receiver that has requested, or will request, its link partner to turn on precoding at the 32.0 GT/s data rate must set the 32.0 GT/s Transmitter Precode Request to 1b in the 32.0 GT/s Status Register ; else it must set the bit to 0b.

Errata: Base 6.3

B820△◀

- A Transmitter that has turned on precoding for the 64.0 GT/s data rate on Lane 0 must set the 64.0 GT/s Transmitter Precoding On bit to 1b in the 64.0 GT/s Status Register ; else it must set the bit to 0b. A Receiver that has requested, or will request, its link partner to turn on precoding at the 64.0 GT/s data rate must set the 64.0 GT/s Transmitter Precode Request to 1b in the 64.0 GT/s Status Register ; else it must set the bit to 0b.
- See § Section 4.2.2.5.1 for 32.0 GT/s precoding requirements. See § Section 4.2.3.1.4 for 64.0 GT/s and above data rate precoding requirements.
- When in Loopback.Active , a Loopback Follower’s Transmitter must not apply any additional Up scrambling or Down scrambling to the looped-back bit stream.

Errata: Base 6.3
B831△◀▷

4.2.2.5.1 Precoding at 32.0 GT/s Data Rate §

When precoding is on at 32.0 GT/s, the following rules apply (see § Figure 4-20):

- Only scrambled bits are precoded.
- The “previous bit” used for precoding is set to 1b on every block boundary and gets updated by the last scrambled and precoded bit transmitted within the current block boundary.
- For symbols that are scrambled, Receivers must first decode the precoded bits before sending them to the descrambler.

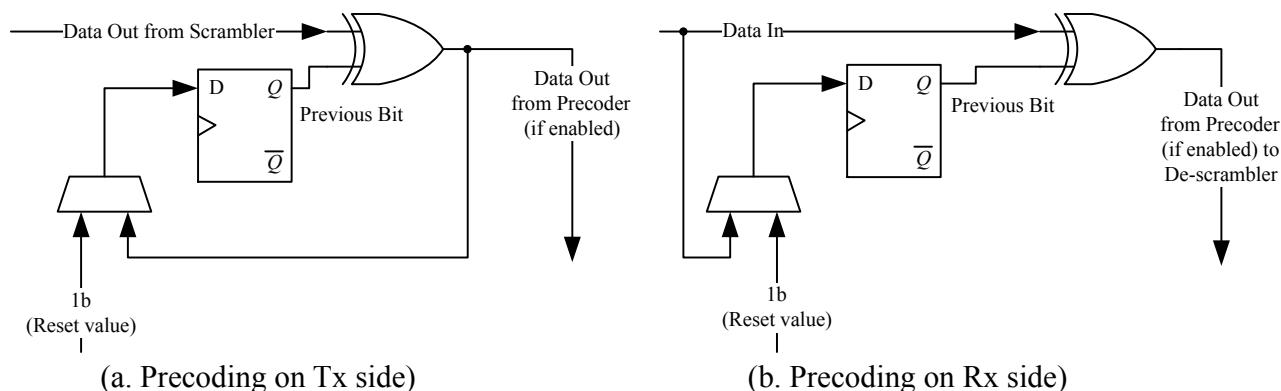


Figure 4-20 Precoding working the scrambler/de-scrambler §

IMPLEMENTATION NOTE: PARITY IN THE SKP ORDERED SET WHEN PRECODING IS TURNED ON §

As per the rules of § Section 4.2.5.1 and § Section 4.2.8.2 , when precoding is turned on, the parity in the SKP Ordered Sets should be calculated before precoding is applied on the Transmit side. Thus, the order in the Transmitter is:

1. scrambling,
2. followed by parity bit calculation,
3. followed by precoding for the scrambled bits.

Accordingly, in the Receiver, the order is:

1. precoding (if turned on by the Transmitter),
2. followed by parity bit calculation,
3. followed by descrambling.

The rationale for this order is that in a Link with one or two Retimers, different Link segments may have the precoding on or off. Let us consider an example system with one retimer between the Root Port and End Point to illustrate this. In the upstream direction, the End Point has precoding on in its Transmitter Lanes since the Retimer Receiver needs it but the Retimer to Root Port Link segment has the precoding off since the Root Port does not need precoding at its Receiver. Since the Retimer does not change the parity bit, the Root Port would get a parity error if the parity calculation was done by the Transmitter (of the End Point) after precoding.

IMPLEMENTATION NOTE: LOOPBACK LEAD'S BEHAVIOR IF PRECODING IS ON IN ANY LINK SEGMENT §

As per the rules of precoding mentioned in this section and § Section 4.2.7.4 , a Loopback Lead operating at a data rate of 32.0 GT/s or higher should account for precoding to be on some link segments and off in other link segments. This is particularly relevant when the Loopback Follower transitions from sending TS1 Ordered Sets to looping back the bits. This is where the precoding on the receiver of the Loopback Lead may switch (between precoding on and off). The Loopback Lead is permitted to use implementation specific mechanisms to handle this scenario.

IMPLEMENTATION NOTE: TS1/TS2 ORDERED SETS WHEN PRECODING IS TURNED ON §

As per the rules in this section, when precoding is turned on, the ‘previous bit’ used for precoding is 1b for the first bit of Symbol 1 since Symbol 0 is not scrambled and the ‘previous bit’ gets set to 1b at the block boundary.

4.2.2.6 Loopback with 128b/130b Code in Non-Flit Mode and Flit Mode §

When using 128b/130b encoding, Loopback Leads must transmit Blocks with the defined 01b and 10b Sync Headers. However, they are not required to transmit an SDS Ordered Set when transitioning from Ordered Set Blocks to Data Blocks, nor are they required to transmit an EDS Token when transitioning from Data Blocks to Ordered Set Blocks. Leads must transmit SKP Ordered Sets periodically as defined in § Section 4.2.8 , and they must be capable of processing received (looped-back) SKP Ordered Sets of varying length. Leads are permitted to transmit Electrical Idle Exit Ordered Sets (EIEOS) as defined in § Section 4.2.2.2.1 . Leads are permitted to transmit any payload in Data Blocks and Ordered Set Blocks that they expect to be looped-back. If the Loopback Lead transmits an Ordered Set Block whose first symbol matches the first symbol of SKP OS, EIEOS , or EOS , that Ordered Set Block must be a complete and valid SKP OS, EIEOS , or EOS .

When using 128b/130b encoding, Loopback Followers must retransmit all bits received without modification, except for SKP Ordered Sets which can be adjusted as needed for clock compensation. If clock compensation is required, Followers must add or remove 4 SKP Symbols per Ordered Set. The modified SKP Ordered Set must meet the definition of § Section 4.2.8.2 (i.e., it must have between 4 to 20 SKP Symbols followed by the SKP-END Symbol and the three Symbols that follow it as transmitted by the Loopback Leads . If a Follower is unable to obtain Block alignment or it is misaligned, it may be unable to perform clock compensation and therefore unable to loop-back all bits received. In this case, it is permitted to add or remove Symbols as necessary to continue operation. Followers must not check for a received SDS Ordered Set when a transition from Ordered Set Blocks to Data Blocks is detected, and they must not check for a received EDS Token when a transition from Data Blocks to Ordered Set Blocks is detected.

4.2.3 Flit Mode Operation §

Flit definition and Symbol placement is specified in this section.

Flits encoded with 8b/10b encoding follow the rules in § Section 4.2.1 with the following exception:

- Since the Flit Mode has its fixed TLP and DLP placement, the packet markers such as STP, SDP, END, EDB are not used.

Flits Encoded with 128b/130b encoding follow the rules in § Section 4.2.2 with the following exceptions:

- Since the Flit Mode has its fixed TLP and DLP placement, the Framing Tokens are not used.

1b/1b encoding is specified in § Section 4.2.3.1 .

4.2.3.1 1b/1b Encoding for 64.0 GT/s and higher Data Rates §

When the PCI Express Link is operating at 64.0 GT/s or higher Data Rates, it uses Flit Mode. A Symbol (8 bits) is the basic unit of transfer per Lane. PAM4 signaling is used for all Symbols, whether it belongs to a Data Stream or an Ordered Set. PAM4 operates on 2-bit aligned boundaries. A Symbol, represented as ‘S₇S₆S₅S₄S₃S₂S₁S₀’, is placed on a Lane starting with the voltage encoding of ‘S₁S₀’ and ending with ‘S₇S₆’. An example placement of Flits (described later) or Ordered Set on a x4 Link is shown in § Figure 4-21 below.

Base 6.4 vs Base 6.3

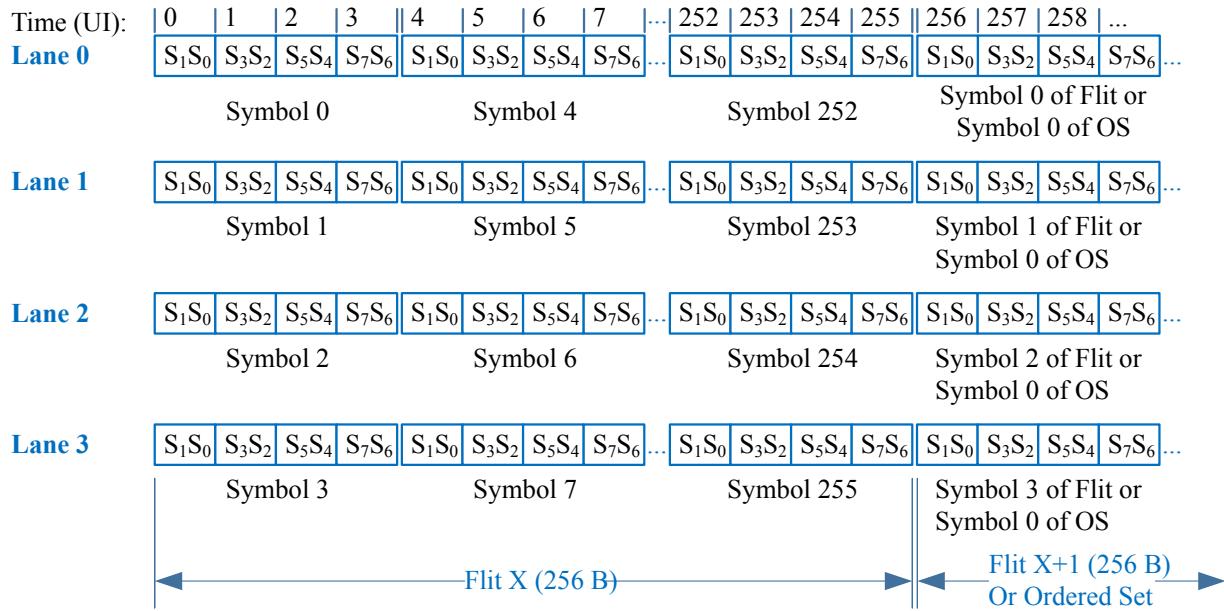


Figure 4-21 Example of Symbol placement in a x4 Link with 1b/1b encoding §

A conceptual diagram of the Transmit side and Receive side is shown in § Figure 4-22 and § Figure 4-23 respectively. These diagrams are provided to explain the order of operations that must be performed on the Transmit and Receive side. An implementation may choose to operate on different widths depending on the design goals and constraints.

At a Flit level, on the Transmit side, CRC is first applied followed by FEC generation. After that, each Lane is allotted its Symbol in the byte interleaved fashion for the Flit. On a per Lane basis, Scrambling is performed, if required, followed by Gray Coding at a 2-bit aligned boundary, after which Precoding is performed at 2-bit aligned UI level, if enabled and required. Thus, in the Symbol above, Gray Coding and Precoding will be applied for the bits corresponding to 'S₁S₀', 'S₃S₂',..., 'S₇S₆'. As shown in the diagram below, the scrambled symbols of the TS1 / TS2 Ordered sets also undergo the precoding logic (including bypass). All Symbols, whether they belong to a Flit (Data Stream) or scrambled Symbols of an Ordered Set or any pattern where explicitly mentioned (e.g., scrambled part of modified compliance pattern) undergo the Gray Coding and PAM4 encoding, whereas unscrambled Symbols of an Ordered Set or any pattern only undergo the PAM4 encoding (the gray coding and DC-balance is effectively taken care of through the bit definition). For the cases where parts of a pattern may be expressed as a repeated sequence of UIs, the Symbol here refers to the aligned 4 UI interval of that pattern.

Base 6.4 vs Base 6.3

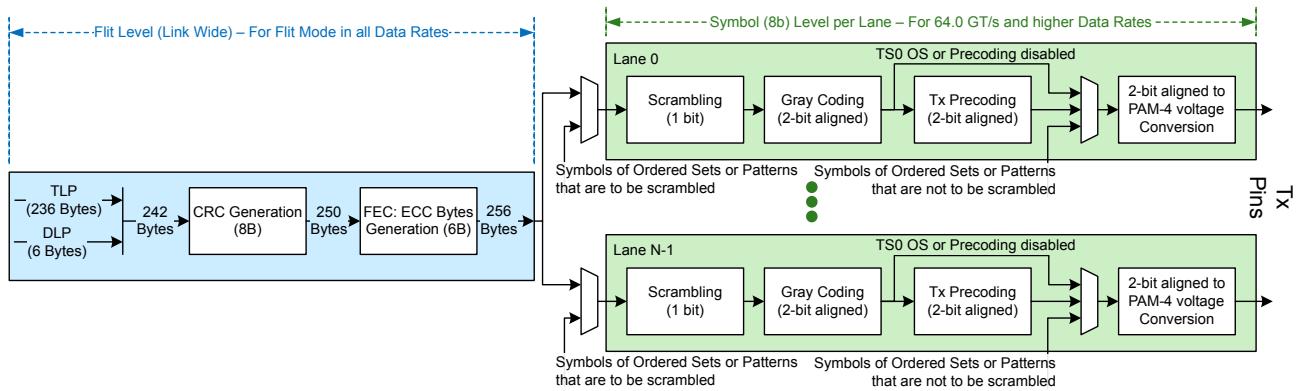


Figure 4-22 Transmit side at 64.0 GT/s §

The Receive side is similar to the Transmit side in the opposite direction. After the PAM4 voltage is converted to a 2-bit aligned quantity, it undergoes the Receive side precoding if applicable followed by the decoding of the Gray code, followed by descrambling on a single bit level, if applicable. The Data Stream (Flit) is aggregated across all Lanes and undergoes the FEC decode and correction followed by the CRC check before sending up the Transactions and Data Link Payload.

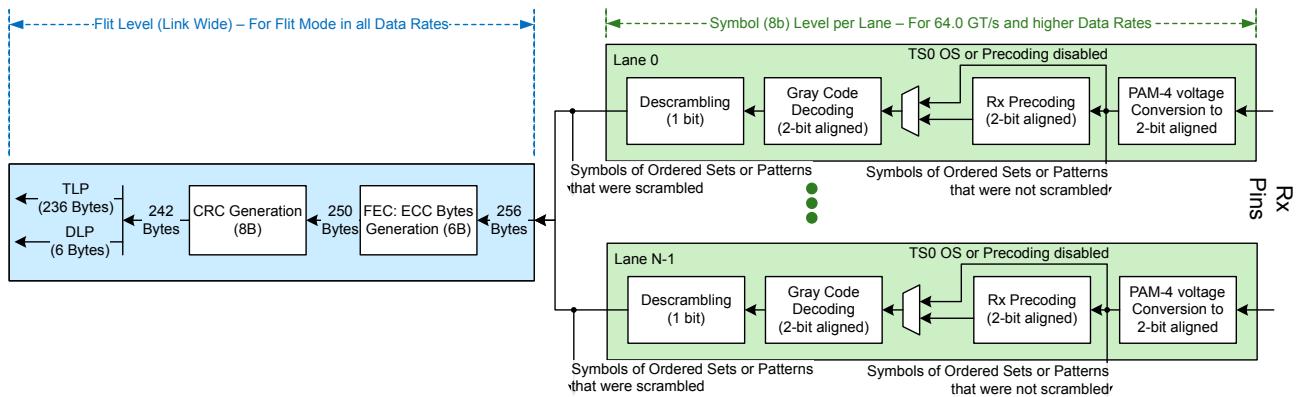


Figure 4-23 Receive side at 64.0 GT/s §

4.2.3.1.1 PAM4 Signaling §

PAM4 stands for Pulse Amplitude Modulation 4-levels. It is a signaling mechanism where 4 levels (2 bits) are encoded in same Unit Interval (UI) resulting in 3 eyes, as shown in § Figure 4-24 . It is deployed only for 64.0 GT/s or higher Data Rates. As described in § Chapter 8. , PAM4 helps with the channel loss as it has the same Nyquist frequency as 32.0 GT/s for 64.0 GT/s Data Rate. The four voltage levels 0, 1, 2, and 3, nominally map to -400 mV, -133 mV, +133 mV and +400 mV respectively, and the Gray code encoded values of 00b, 01b, 11b, and 10b respectively, with the little-endian bit order, as shown in § Figure 4-24 . The corresponding DC Balance values to be used when designing Ordered Sets to meet the DC

balance needs is also shown in § Figure 4-24 . The Reduced voltage levels (eye height or EH) and eye width (EW) increases susceptibility to errors. Gray coding is used to help minimize errors within a UI for the voltage levels.

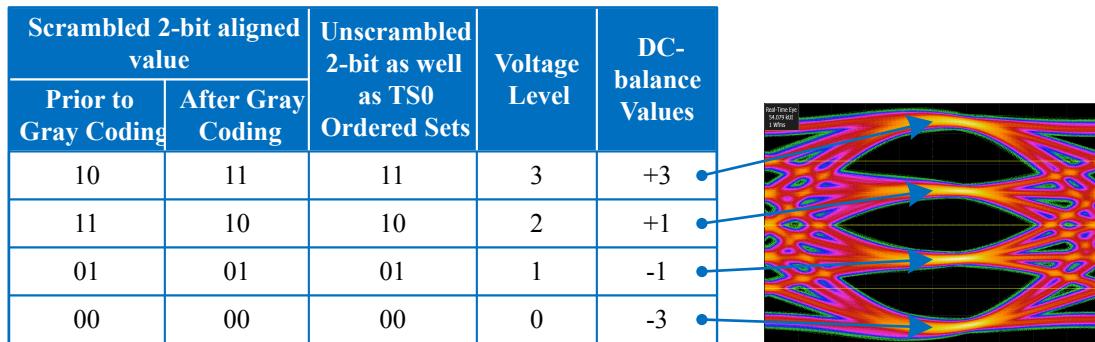


Figure 4-24 PAM4 Signaling at UI level: Voltage levels, 2-bit encoding, and their corresponding DC balance values §

With PAM4 encoding, the bit error rate (BER) is expected to be significantly worse than the 10^{-12} BER target of the lower data rates (2.5, 5.0, 8.0, 16.0, and 32.0 GT/s). In addition, errors are expected to occur in bursts in a Lane and some amount of Lane to Lane correlation is also expected. The electrical spec parameters along with FBER (First Bit Error Rate) $< 10^{-6}$ described in § Chapter 8. must be met to ensure probability of a Flit error after FEC to be less than 3×10^{-5} .

A **Forward Error Correction (FEC)** mechanism is used to deal with the high FBER in the Data Stream. Since FEC works on fixed sized code words, Flit (flow control unit) will be used for sending/ receiving TLPs and DLLPs in a data stream. A low-overhead FEC will be used to keep the latency low. This will be augmented with a strong CRC at Flit level for high reliability. Link level retry will be deployed at the Flit level in Flit Mode. Ordered Sets will be protected through replication for Data Rates of 64.0 GT/s and above. Additionally, the precoding mechanism described below will be used for all scrambled bits to help minimize the number of errors in an error burst within a Lane.

4.2.3.1.2 1b/1b Scrambling §

The scrambling mechanism in the 64.0 GT/s Data Rate is identical to that for the 8.0, 16.0, and 32.0 GT/s Data Rates and is already covered in § Section 4.2.2.4 , except for the TS0 / TS1 / TS2 / SKP Ordered Set rules defined below. § Figure 4-25 demonstrates the scrambler on the Transmitter side. All Data Stream bits as well as some Ordered Set bits are scrambled.

- TS1 and TS2 Ordered Sets:
 - Symbols 0 and 8 bypass scrambling.
 - Symbols 1 through 6 and Symbols 9 through 14 are scrambled.
 - Symbols 7 and 15 bypass scrambling if required for DC Balance, but they are scrambled if not required for DC Balance.
- TS0 Ordered Sets:
 - Symbols 0 and 8 bypass scrambling
 - Symbols 1 through 6 and Symbols 9 through 14 use NRZ-based scrambling
 - NRZ-based scrambling consists of scrambling all 8 bits and then, in place of gray-coding, forcing the even bit $2i$ to be identical to the odd bit $(2i+1)$ [where $0 \leq i \leq 3$]. This ensures that only voltage levels 0 and 3 are sent.

- Symbols 7 and 15 bypass scrambling if required for DC Balance; else they carry even parity of Symbols 1 through 6 and Symbols 9 through 14 respectively with NRZ-scrambling
- All Symbols in TS0 are NRZ-based (i.e., each UI maps to voltage levels 0 or 3, encoded as 00b or 10b). Only TS0 Ordered Set is designed to be NRZ-based in 64.0 GT/s Data Rate.
- SKP Ordered Sets :
 - When processing Ordered Sets outside of a Data Stream, receivers use the 1b/1b SKP decode rules in § Section 4.2.3.1.5 . When processing Ordered Sets inside a Data Stream, receivers use the 1b/1b SKP decode rules in § Section 4.2.3.2 . If it is determined by the Receiver that the Ordered Set is a SKP Ordered Set , then the LFSR is not advanced for any Symbol of the Block, as specified in § Section 4.2.2.4 .

IMPLEMENTATION NOTE: NRZ-BASED SCRAMBLING §

During NRZ-based scrambling, forcing the even bit $2i$ to be identical to the odd bit $(2i+1)$ [where $0 \leq i \leq 3$] can be performed after gray-coding or instead of gray-coding. The result will be the same either way.

4.2.3.1.3 Gray Coding at 64.0 GT/s and Higher Data Rates §

Only scrambled bits on a 2-bit aligned boundary are Gray coded, when both bits are scrambled. All bits in the Data Stream are scrambled and Gray coded. Only some Symbols of the TS1 / TS2 Ordered Sets are scrambled – those are Gray coded, while the unscrambled symbols are not Gray coded. All TS0 Ordered Set Symbols to be Half Scrambled undergo NRZ-based scrambling, as described in § Section 4.2.3.1.2 . § Figure 4-25 represents the sequence of Gray Coding, which is followed by Tx Precoding, when enabled, followed by the PAM4 voltage translation on the Tx side as well as the Rx side, considering an error E_n that may affect the nth 2-bit quantity on the channel.

Gray Coding is only enabled in the PAM4 mode (at 64.0 GT/s and above Data Rates). Gray-coding works on aligned 2-bit quantities on a per-Lane basis. On the Transmit side, input G_n ($G_{n1} G_{n0}$) is transformed to output P_n ($P_{n1} P_{n0}$), using the Gray code encoding where the input 00b, 01b, 10b, 11b becomes 00b, 01b, 11b, and 10b respectively in the output. This can be represented by the equations: $P_{n1} = G_{n1}$ and $P_{n0} = G_{n1} \wedge G_{n0}$. The Receive side is identical and can be represented as: $G'_{n1} = P'_{n1}$ and $G'_{n0} = P'_{n1} \wedge P'_{n0}$. § Table 4-6 demonstrates how a likely error of ± 1 on voltage level at most affects one bit with Gray Coding.

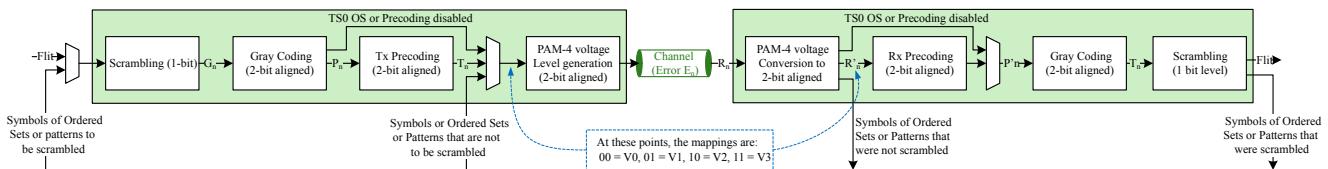


Figure 4-25 The Sequence of Gray Coding, Precoding, and PAM4 voltage translation on an aligned 2-bit boundary on a per Lane §

Table 4-6 Effect of +/-1 voltage level error on the wire for various PAM4 voltage levels – at most one bit flips with an error on a UI §

| Voltage Level | Resulting Voltage Level | |
|---------------|-------------------------|--------------|
| | Error of +1 | Error of -1 |
| 0 | 1 | 0 (no error) |
| 1 | 2 | 0 |
| 2 | 3 | 1 |
| 3 | 3 (no error) | 2 |

4.2.3.1.4 Precoding at 64.0 GT/s and Higher Data Rates §

Only scrambled bits on a 2-bit aligned boundary are precoded, when both bits are scrambled and precoding is enabled. Precoding is bypassed for all Symbols of TS0 Ordered Sets. Other than the precoding mechanism described here, all the precoding rules in § Section 4.2.2.5 apply to 1b/1b encoding with the exception of TS0 Ordered Set. § Figure 4-25 represents the Precoding on the Transmit as well as Receive side. On the Transmit side, the input is P_n and the output is T_n . T_{n-1} represents the output from the precoder in the previous UI. When the prior UI was unscrambled and precoding is enabled, T_{n-1} is set to 00b. The output T_n will be converted to the PAM4 voltage levels described above, when precoding is enabled. The channel may inject an error E_n (no error means $E_n = 00b$). On the Receive side, $R_n = T_n + E_n$. The input to the Precoding logic is R'_n ($R_n = T_n + e_n$), where e_n is inclusive of the error on the wire E_n as well as any internal DFE propagated error. The output of the precoding logic is P'_n , when precoding is enabled. The Truth Table for the Precoding function for the Transmit and Receive side is shown in § Table 4-7 and § Table 4-8 . Precoding can help mitigate the number of errors in a burst.

Table 4-7 Truth Table for Precoding on the Transmit side §

| | | T_n | | | |
|-------|----|-----------|----|----|----|
| | | T_{n-1} | | | |
| P_n | | 00 | 01 | 11 | 10 |
| | | 00 | 11 | 01 | 10 |
| 01 | 01 | 00 | 10 | 11 | |
| 11 | 11 | 10 | 00 | 01 | |
| 10 | 10 | 01 | 11 | 00 | |

Table 4-8 Truth Table
for Precoding on the
Receive side §

| | | P' n | | | |
|----|----|--------|----|----|----|
| | | R' n-1 | | | |
| | | 00 | 01 | 11 | 10 |
| 00 | 00 | 00 | 01 | 11 | 10 |
| 01 | 01 | 01 | 10 | 00 | 11 |
| 11 | 11 | 11 | 00 | 10 | 01 |
| 10 | 10 | 10 | 11 | 01 | 00 |

IMPLEMENTATION NOTE: HOW DOES PRECODING HELP REDUCE THE NUMBER OF ERRORS? §

Precoding is effective with Gray coding on certain channels to reduce the impact of errors in the presence of DFE. It works similar to the precoding in 32.0 GT/s, except here it works on 2-bit aligned quantities with PAM4 encoding. The most likely error scenario is $+/-1$ on the voltage level over one UI with PAM4 signaling. § Table 4-6 demonstrates the resulting voltage level under this error scenario and the resulting bit error with Gray code encoding. Gray Coding results in at most a single bit flip within that UI. Error propagation due to DFE will most likely occur in consecutive UIs. Under these assumptions, precoding ensures that the error appears in two bits within a wire: when the error gets introduced in the wire and the UI after the DFE burst stops. For cases such as error voltage magnitude is $> +1$ or < -1 or for cases where the DFE introduces errors in non-contiguous UI's, precoding may not be effective.

The precoding equation on Transmit side, based on the Truth Table in § Table 4-7 is: $T_n = (P_n - T_{n-1}) \bmod 4$, which is equivalent to $P_n = (T_n + T_{n-1}) \bmod 4$. The precoding equation on the Receive side, based on the Truth Table in § Table 4-8 is: $P'_n = (R_n + R_{n-1}) \bmod 4$. This can be simplified as: $P'_n = (T_n + e_n + T_{n-1} + e_{n-1}) \bmod 4 = (P_n + e_n + e_{n-1}) \bmod 4$. When we have the first error burst on the wire, $E_n = +/-1$ which will translate to $e_n = E_n (+/-1)$. So, with FBER, one bit gets affected in the UI where the external error happened. In the subsequent UIs, in the absence of any additional external errors, $e_n = -e_{n-1}$ or 0 (0 when DFE stops propagating the error, when it does propagate the magnitude is negative of the earlier error since the corresponding tap has a negative weight). If the error propagates due to DFE, $e_n + e_{n-1} = 0$. If it stops propagating, e_n becomes $+/-1$. So only two bit errors will be observed when precoding is effective: first bit error in the UI where the first error occurred in the wire and the UI following when DFE stopped propagating the error burst.

§ Table 4-9 illustrates the concept with an error in the channel and subsequent DFE error propagations. The bits after scrambling to be sent across the Link across successive UIs are in the first row, represented as G_n (pre-gray coding). The different stages are represented in subsequent rows in the table. It must be noted that after gray coding (starting with T_n) until we decode with the gray-coding at the Receiver (G'_n), we are dealing with voltage levels of 0, 1, 2, 3. The channel error occurs with the PAM-4 Symbol in UI #1 of magnitude -1, which causes only one error of magnitude in the Receiver pin (R_n). However, due to DFE burst, the error propagates in the Receiver circuits from UI# 1 through 5 (R'_n). Once we precode in the Receiver, these errors occur in two places, UI #1 and 6, as shown in P'_n , which then translates to two bit errors, one each in UI #1 and 6, as shown in G'_n .

Table 4-9 Example of precoding with an error in the channel and DFE error propagation at the Receiver §

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----------------------|-------------------------------------|----|----|----|----|----|----|----|----|
| Voltage Level Domain | G_n | 10 | 11 | 10 | 11 | 00 | 01 | 00 | 01 |
| | P_n | 11 | 10 | 11 | 10 | 00 | 01 | 00 | 01 |
| | $T_n^{(+ (P_n - T_{n-1}) \bmod 4)}$ | 3 | 3 | 0 | 2 | 2 | 3 | 1 | 0 |
| | E_n (Channel Error) | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R_n | 3 | 2 | 0 | 2 | 2 | 3 | 1 | 0 |
| | DFE Error | 0 | 0 | +1 | -1 | +1 | -1 | 0 | 0 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|----|----|----|----|----|----|----|----|
| e_n (Net Error: $E_n - DFE\ Error$) | 0 | -1 | +1 | -1 | +1 | -1 | 0 | 0 |
| R'_n (Difference from T_n is an error) | 3 | 2 | 1 | 1 | 3 | 2 | 1 | 0 |
| P'_n ($+ (R_n - R_{n-1}) \bmod 4$) | 3 | 1 | 3 | 2 | 0 | 1 | 3 | 1 |
| G'_n (Difference from G_n is an error) | 10 | 01 | 10 | 11 | 00 | 01 | 10 | 01 |

4.2.3.1.5 Ordered Set Blocks at 64.0 GT/s and Higher Data Rates §

All Ordered Sets with the exception SKP Ordered Sets (i.e., TS0, TS1, TS2, EIOS, EIEOS and SDS) are 16 Bytes long on both the Transmit and the Receive side. SKP Ordered Sets are 40 Bytes long on the Transmit side and can be 24, 32, 40, 48 or 56 bytes long on the Receive side. Ordered Sets exhibit a greater degree of redundancy than the corresponding Ordered Sets at Data Rates lower than 64.0 GT/s. The redundancy is required due to the high FBER with correlated errors with PAM4 signaling. The Ordered Sets are described in detail in § Section 4.2.5.1, § Section 4.2.5.3, § Section 4.2.5.7, and § Section 4.2.8.3.

While processing Ordered Sets outside of a Data Stream, the following rules apply for the EIEOS, SKP OS, and EIOS, after obtaining Block alignment:

- An EIEOS is considered to be received under the following conditions:
 - any 5 or more aligned and consecutive bytes in each of the first and last 8 bytes of a Block match the corresponding bytes of an EIEOS and
 - either Symbol 0 or Symbol 8 of the Block matches the corresponding byte of an EIEOS. A Receiver is permitted to ignore up to one bit mismatch for this comparison

While in Aligned phase, a Receiver must switch the block boundary by > 1 UI on an EIEOS Ordered Set or should shift the Block boundary by ≤ 1 UI only if it receives an EIEOS followed by a valid first symbol of an expected Ordered Set.

While searching for an EIEOS, the 8-byte boundary can begin in any aligned UI and the bits must be looked at prior to performing gray coding, precoding, or descrambling in the Receiver side.

- An EIOS is considered to be received under the following conditions:
 - any 5 or more aligned bytes of the first 8 bytes of a Block match the corresponding bytes of an EIOS and
 - either Symbol 0 or Symbol 8 of the Block matches the corresponding byte of an EIOS. When an EIOS is received, the Lane is ready to enter Electrical Idle irrespective of what is received in the last 7 Bytes of the Block.
- A SKP OS is considered to begin under the following conditions: (i) any 5 or more aligned bytes of the first 8 bytes of a Block match a SKP, and (ii) either Symbol 0 is a SKP or Symbol 8 of the Block is either a SKP or a SKP-END. It looks at each subsequent aligned 8 byte chunks and applies the following rules:
 - If any 5 or more aligned bytes match SKP-END or the current chunk is the fifth 8 byte chunk after the start of the SKP OS (i.e., current set is bytes 40 through 47 from the start of the SKP OS), the SKP OS will terminate after the next aligned 8 byte chunk.

IMPLEMENTATION NOTE: INFERRING ORDERED SETS WITHIN AND OUTSIDE OF DATA STREAMS §

Since most bytes of TS0 / TS1 / TS2 Ordered Sets are scrambled (except Symbols 0 and 8), one has to get an exact match in at least Symbol 0 or 8, in addition to the match in 5 Byte positions to infer the EIEOS, EIOS, or SKP OS, while not in a Data Stream. The rules for detection of TS0 , TS1 , TS2 , and SDS Ordered Set appears with the description of the Ordered Set. Since SKP, EIEOS , and EIOS can appear within a Data Stream, there are slightly different rules for inferring those in a Data Stream than while not in a Data Stream, taking advantage of the fact that none of these Ordered Sets have any scrambling, to ensure higher fault tolerance during a Data Stream. Hence the rules for detection of EIOS , EIEOS and SKP OS are specified twice; in this section as well as in § Section 4.2.8.3 .

4.2.3.1.6 Alignment at Block/ Flit Level for 1b/1b Encoding §

With 1b/1b encoding, Block alignment aligns with the Ordered Set boundary on a per-Lane basis. Once this is accomplished, the Flit-level alignment at the Link level occurs automatically with the Data Stream that starts at the end of the Control SKP Ordered Set that immediately follows the SDS Ordered Set sequence. Once the Flit alignment starts, it adjusts to the Flit boundary. When an Ordered Set is scheduled to be received and the received Ordered Set is a Control SKP Ordered Set , the Flit boundary is automatically adjusted to start at the conclusion of the Ordered Set.

The Block level alignment occurs with the EIEOS Ordered Set in Configuration or Recovery states. The EIEOS is a unique pattern that Receivers use to determine the start/ end of the Ordered Set boundary in the received bit stream. Conceptually, Receivers can be in three different phases of alignment: Unaligned, Aligned, and Locked. These phases are defined to illustrate the required behavior, but are not meant to specify a required implementation.

Unaligned Phase

Receivers enter this phase after a period of Electrical Idle, such as when the data rate is changed to one that uses 1b/1b encoding or when they exit a low-power Link state, or if directed (by an implementation specific method). In this phase, Receivers monitor the received bit stream for a match against all 128 bits of the EIEOS bit pattern. When one is detected, they adjust their alignment to it and proceed to the Aligned Phase.

Aligned Phase

Receivers monitor the received bit stream for the EIEOS OS and the received Blocks for a Start of Data Stream (SDS) Ordered Set. If an EIEOS bit pattern is detected on an alignment that does not match the current alignment and the LTSSM is in the Recovery.RcvrLock state, Receivers must adjust their alignment to the newly received EIEOS bit pattern. If an EIEOS bit pattern is detected on an alignment that does not match the current alignment, the LTSSM is in the Recovery.RcvrCfg state, and the subsequent Symbol matches the first Symbol of an expected Ordered Set, Receivers must adjust their alignment to the newly received EIEOS bit pattern. If an SDS Ordered Set is received, Receivers proceed to the Locked phase. The Data Stream starts after the SDS Ordered Set sequence is received, though Flits start after the Control SKP Ordered Set immediately succeeding the SDS Ordered Set sequence. The conclusion of the SKP Ordered Set immediately following an SDS Ordered Set sequence marks the start of the Flit boundary at the end of the Control SKP Ordered Set .

Locked Phase

Receivers must not adjust their Block or Flit alignment while in this phase. Data Blocks are expected to be received after the Control SKP Ordered Set following the SDS Ordered Set sequence, and adjusting the Block alignment would interfere with the processing of these Blocks. Receivers must return to the Unaligned or Aligned Phase if the Data Stream is terminated by the Link entering Recovery and/or a Framing Error is detected.

Additional Requirements:

- While in the Aligned Phase, Receivers must adjust their Block alignment, as necessary, when a Control SKP Ordered Set is received. See § Section 4.2.8 for more information on SKP Ordered Sets.
- While in the Locked phase, Receivers must adjust their Block and Flit alignment, as necessary, when a Control SKP Ordered Set is received.
- After any LTSSM transition to Recovery , Receivers must ignore all received TS Ordered Sets until they receive an EIEOS . Conceptually, receiving an EIEOS validates the Receiver's alignment and allows TS Ordered Set processing to proceed. If a received EIEOS initiates an LTSSM transition from L0 to Recovery , Receivers are permitted to process any TS Ordered Sets that follow the EIEOS or ignore them until another EIEOS is received after entering Recovery .
- Receivers are permitted to transition from the Locked phase to the Unaligned or Aligned phase as long as Data Stream processing is stopped.
- Loopback Leads : While in Loopback.Entry , Leads must be capable of adjusting their Receiver's Block / Flit alignment to received EIEOS bit patterns. While in Loopback.Active , Leads are permitted to transmit an EIEOS and adjust their Receiver's Block alignment to the looped back bit stream.
- Loopback Followers : While in Loopback.Entry , Followers must be capable of adjusting their Receiver's Block alignment to received EIEOS bit patterns. While in Loopback.Active , Followers must not adjust their Receiver's Block alignment, except at the scheduled Control SKP Ordered Set boundary. Conceptually, the Receiver is directed to the Locked phase when the Follower starts to loop back the received bit stream.

4.2.3.2 Processing of Ordered Sets During Flit Mode Data Stream §

Flit Mode does not employ the EDS ('End of Data Stream') token. Instead, a Data Stream can be terminated by either an EOS (prior to entering a low power state, such as L1) or an EIEOS (upon entry into Recovery). SKP Ordered Sets appear at periodic intervals during the Data Stream (as defined in § Table 4-32). When necessary, the EOS or EIEOS must be transmitted in place of the SKP Ordered Set , as defined in § Section 4.2.3.4.6 . Therefore, these three Ordered Sets are defined such that one cannot alias to another even with a burst error.

Ordered Set processing during or at the end of a Data Stream must be performed as follows by the Receiver for 64.0 GT/s or higher Data Rates:

- Receiver checks the first 8 bytes of the Ordered Set in the place where an Ordered Set is expected to appear.
- Any 5 or more aligned bytes of the first aligned 8 byte chunk matches the corresponding bytes of either a SKP, EOS , or EIEOS for the corresponding Ordered Set to be considered valid in each of the active Lanes. Failure to meet this requirement is considered a framing error and the Link must enter Recovery .
- If a SKP Ordered Set is inferred from the first 8 bytes, the Receiver looks at each subsequent aligned 8 byte chunks and applies the following rules:
 - If at least each of any 5 aligned bytes matches SKP-END or the current chunk is the fifth 8 byte chunk after the start of the SKP OS (i.e., current set is bytes 40 through 47), the SKP OS will terminate after the next aligned 8 byte chunk. At the conclusion of the SKP OS the Data Stream resumes, if the SKP OS terminated with the receipt of 5 aligned bytes of SKP-END. If the SKP OS is being terminated without receipt of 5 aligned bytes of SKP-END, a Framing Error occurs.
- Any of the following conditions are Framing Error even with the receipt of a proper OS on each of active Lanes and the Link must enter Recovery :
 - Receiving any combination of these ordered sets simultaneously on any two active Lanes: EIEOS and EOS or EIEOS and SKP OS,
 - Receiving a SKP OS of unequal lengths across all Lanes
 - Receiving a combination of EOS and SKP OS and any of the following conditions is true:

- L0p has not been enabled in the Link
 - The set of Lanes receiving EIOS are not contiguous
 - The set of Lanes receiving SKP OS are not contiguous
 - The number of Lanes receiving SKP OS is not a valid link width (1, 2, 4, or 8)
 - Any of these Lanes is receiving neither SKP OS nor EIOS.
- If an EIEOS is inferred from the first 8 bytes on any active Lane, the Data Stream ends with the receipt of the EIEOS and the Link must enter Recovery when permitted by the LTSSM.
 - If an EIOS is inferred from the first 8 bytes across any active Lanes, the Link prepares to enter the low-power state for which the negotiation has occurred (L0p or L1 or L2).
 - Data stream continues on a reduced set of Lanes at the conclusion of the OS if all of the following conditions are true:
 - An EIOS is inferred from the first 8 bytes on some active Lanes,
 - The remaining active Lanes received the SKP OS on 1, 2, 4 or 8 Lanes, and
 - L0p has been enabled in the Link

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: 64.0 GT/S DATA STREAM ORDERED SET PROCESSING §

A Flow chart depicting part of the above flow is illustrated in § Figure 4-26 . This flow chart is not meant to specify all the rules but illustrate some of the rules for illustration purposes.

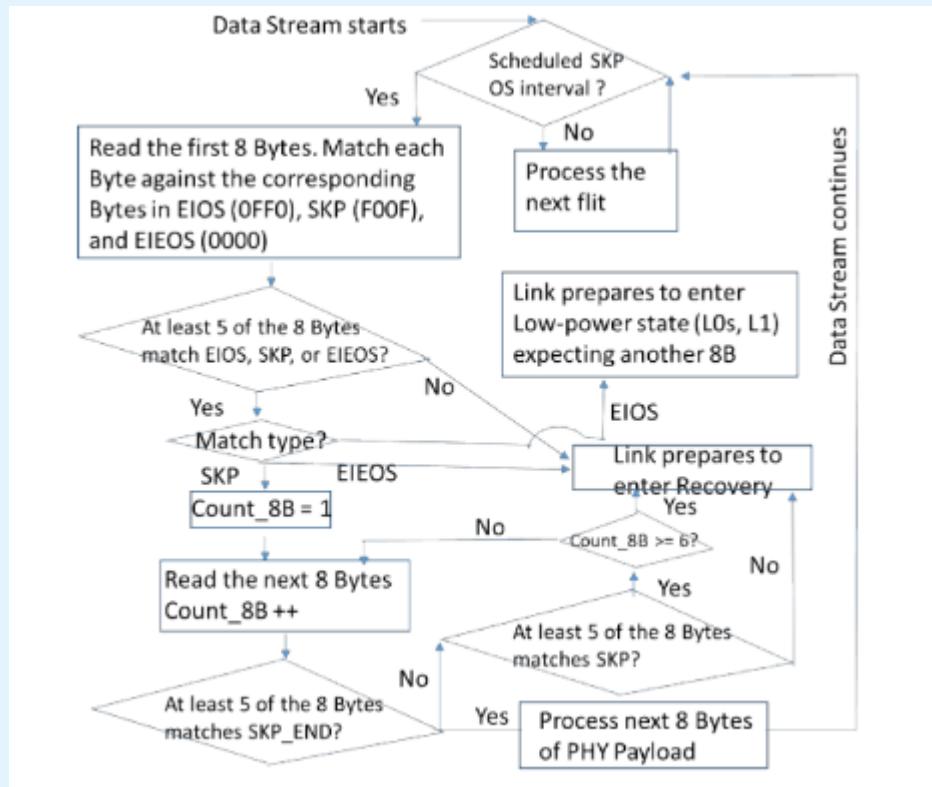


Figure 6-14 Processing flit-encoding Ordered Sets during or at the end of a Data Stream with flit encoding

Figure 4-26 Processing of Ordered Sets during or at the end of a Data Stream in Flit ↓↓mode↓ ↑↑Mode↑ at 64.0 GT/s Data Rate §

4.2.3.3 Data Stream in Flit Mode §

The flit size is 256 Bytes, allocated as follows:

- 236 Bytes for TLPs (Bytes 0 through 235 in the Flit, as shown in § Table 4-10 through § Table 4-14)
- 6 Bytes for Data Link Layer Payload (DLP in Bytes 236 through 241 in the Flit, shown as DLP0..6 in § Table 4-10 through § Table 4-14)
- 8 Bytes for CRC (Bytes 242 through 249 in the Flit, shown as CRC0..7 in § Table 4-10 through § Table 4-14), and
- 6 Bytes for ECC (Bytes 250 through 255 arranged as 3 groups of ECC0[0:1], ECC1[0:1], ECC2[0:1], as shown in § Table 4-10 through § Table 4-14).

A flit is interleaved across the Lanes in the Link in a Byte aligned fashion, as shown in § Table 4-10 through § Table 4-14 . Each Byte corresponds to a Symbol in the 64.0 GT/s and above Data Rates, a Symbol within the Data Block with 128b/

130b encoding, for 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s; and gets converted to a 10b Symbol using the 8b/10b encoding for 2.5 GT/s and 5.0 GT/s Data Rates. Since the TLP and DLP have fixed allocation of bytes within each flit, the Port will schedule the packets of each type accordingly. The 8 Bytes of CRC protects the TLP and DLLP Bytes (and not the ECC bytes). The 6 Bytes of ECC protects the entire flit, including the CRC. Even though the ECC protection is needed only with the high FBER with the PAM4 signaling at 64.0 GT/s and higher Data Rates, it will be deployed for the lower Data Rates for consistency.

On a Link layer retry, only the TLPs will be replayed, and not the DLP. Thus, the link layer payload in each flit (DLP) does not enter the Link Layer Retry Buffer (LLRB) at the Transmitter. The DLP fields in the retried flits will reflect the latest Data Link Layer Payload. Thus, a DLP can be lost and the same mechanism of replication/ aggregation for DLLPs ensures that the data link layer messages gets delivered to the Link partner.

The FEC is a 3-way interleaved ECC, with each ECC code capable of correcting a single Byte error. The interleaving is done so that a burst error of up to 16 bits in any Lane does not impact more than a Byte in each interleaved ECC code word. Each of the interleaved ECC has a different color. Thus, all blue colored Bytes [e.g., 0, 3, 6, 9,..., 231, 234, DLP1, DLP4, CRC1, CRC4, and CRC7] are part of ECC Group 0 and are covered by the 2 blue colored ECC0[0] and ECC0[1] bytes (which are placed in the last Symbol of the flit in Lanes 12 and 15 respectively).

Base 6.4 vs Base 6.3

| Description | | Lane | | | | | | | | | | | | | | | |
|-----------------------|------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 Symbol times | TLP Bytes [0...223] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| | | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| | | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| | | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| | | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| | | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| | | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| | | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| | | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| | | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| | | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| | | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| | | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| | TLP, DLP | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | DLP 0 | DLP 1 | DLP 2 | DLP 3 |
| | DLP, CRC, ECC | DLP 4 | DLP 5 | CRC 0 | CRC 1 | CRC 2 | CRC 3 | CRC 4 | CRC 5 | CRC 6 | CRC 7 | ECC1 [0] | ECC2 [0] | ECC0 [1] | ECC1 [1] | ECC2 [1] | ECC0 [1] |

IMPLEMENTATION NOTE:

The first 236 Bytes (0..235) are for TLP(s), next 6 Bytes are for Data Link Layer Payload (DLP0..5), next 8 Bytes for CRC (CRC0..7) and the last 6 Bytes are for ECC (ecc 0..1). The FEC is a 3-way interleaved ECC, each capable of correcting a single Byte, with the interleaving shown in 3 colors.

Base 6.4 vs Base 6.3

Table 4-11 Flit interleaving in a x8 Link §

| Description | | Lane | | | | | | | | |
|--|-----------------|-------|-------|----------|----------|----------|----------|----------|----------|--|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| TLP Bytes [0...231] | 32 Symbol times | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | |
| | | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | |
| | | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | |
| | | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | |
| | | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | |
| | | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | |
| | | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | |
| | | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | |
| | | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | |
| | | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | |
| | | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | |
| | | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | |
| | | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | |
| | | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | |
| | | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | |
| | | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | |
| | | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | |
| | | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | |
| | | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | |
| | | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | |
| | | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | |
| | | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | |
| | | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | |
| | | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | |
| | | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | |
| | | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | |
| | | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | |
| TLP Bytes [231...235], DLP Bytes | | 232 | 233 | 234 | 235 | DLP 0 | DLP 1 | DLP 2 | DLP 3 | |
| DLP, CRC Bytes | | DLP 4 | DLP 5 | CRC 0 | CRC 1 | CRC 2 | CRC 3 | CRC 4 | CRC 5 | |
| CRC, ECC Bytes | | CRC 6 | CRC 7 | ECC1 [0] | ECC2 [0] | ECC0 [0] | ECC1 [1] | ECC2 [1] | ECC0 [1] | |

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE:

S Note that the same set of Bytes form the same ECC group in the 3-way interleaved FEC as in a x16 Link. The first 236 Bytes (0..235) are for TLP(s), next 6 Bytes are for Data Link Layer Payload (DLP0..5), next 8 Bytes for CRC (CRC0..7) and the last 6 Bytes are for ECC (ecc 0..1). The FEC is a 3-way interleaved ECC, each capable of correcting a single Byte, with the interleaving shown in 3 colors.

| Description | | Lane | | | |
|---------------------|---------------------|----------|----------|----------|----------|
| | | 0 | 1 | 2 | 3 |
| 64 Symbol times | TLP Bytes [0...235] | 0 | 1 | 2 | 3 |
| | | 4 | 5 | 6 | 7 |
| | | 8 | 9 | 10 | 11 |
| | | : | : | : | : |
| | | 224 | 225 | 226 | 227 |
| | | 228 | 229 | 230 | 231 |
| | | 232 | 233 | 234 | 235 |
| | | DLP 0 | DLP 1 | DLP 2 | DLP 3 |
| | | DLP 4 | DLP 5 | CRC 0 | CRC 1 |
| | | CRC 2 | CRC 3 | CRC 4 | CRC 5 |
| DLP, CRC, ECC Bytes | CRC 6 | CRC 6 | CRC 7 | ECC1 [0] | ECC2 [0] |
| | | ECC0 [0] | ECC1 [1] | ECC2 [1] | ECC0 [1] |

IMPLEMENTATION NOTE:

S Note that the same set of Bytes form the same ECC group in the 3-way interleaved FEC as in a x16/x8 Link. The first 236 Bytes (0..235) are for TLP(s), next 6 Bytes are for Data Link Layer Payload (DLP0..5), next 8 Bytes for CRC (CRC0..7) and the last 6 Bytes are for ECC (ecc 0..1). The FEC is a 3-way interleaved ECC, each capable of correcting a single Byte, with the interleaving shown in 3 colors.

| Description | | Lane | |
|------------------|---------------------|-------|-------|
| | | 0 | 1 |
| 128 Symbol times | TLP Bytes [0...235] | 0 | 1 |
| | | 2 | 3 |
| | | 4 | 5 |
| | | : | : |
| | | 230 | 231 |
| | | 232 | 233 |
| | | 234 | 235 |
| | | DLP 0 | DLP 1 |

| Description | Lane | |
|-------------|----------|----------|
| | 0 | 1 |
| CRC Bytes | DLP 2 | DLP 3 |
| | DLP 4 | DLP 5 |
| | CRC 0 | CRC 1 |
| | CRC 2 | CRC 3 |
| | CRC 4 | CRC 5 |
| | CRC 6 | CRC 7 |
| | ecc1 [0] | ecc2 [0] |
| ECC Bytes | ecc0 [0] | ecc1 [1] |
| | ecc2 [1] | ecc0 [1] |

IMPLEMENTATION NOTE:

Note that the same set of Bytes form the same ECC group in the 3-way interleaved FEC as in a x16/x8/x4 Link. The first 236 Bytes (0..235) are for TLP(s), next 6 Bytes are for Data Link Layer Payload (DLP0..5), next 8 Bytes for CRC (CRC0..7) and the last 6 Bytes are for ECC (ecc 0..1). The FEC is a 3-way interleaved ECC, each capable of correcting a single Byte, with the interleaving shown in 3 colors.

| Table 4-14 Flit arrangement in a x1 Link § | |
|--|--|
| Description | Lane |
| | 0 |
| TLP Bytes [0..235] | 0 1 2 ⋮ 230 231 232 233 234 235 |
| 256 Symbol times | DLP 0 DLP 1 DLP 2 DLP 3 DLP 4 DLP 5 |
| CRC Bytes | CRC 0 CRC 1 CRC 2 CRC 3 CRC 4 |

Base 6.4 vs Base 6.3

| Description | Lane |
|-------------|----------|
| | 0 |
| ECC Bytes | CRC 5 |
| | CRC 6 |
| | CRC 7 |
| | ECC1 [0] |
| | ECC2 [0] |
| | ECC0 [0] |
| | ECC1 [1] |
| | ECC2 [1] |
| | ECC0 [1] |

IMPLEMENTATION NOTE:

Note that the same set of Bytes form the same ECC group in the 3-way interleaved FEC as in a x16/x8/x4/x2 Link. The first 236 Bytes (0..235) are for TLP(s), next 6 Bytes are for Data Link Layer Payload (DLP0..5), next 8 Bytes for CRC (CRC0..7) and the last 6 Bytes are for ECC (ecc 0..1). The FEC is a 3-way interleaved ECC, each capable of correcting a single Byte, with the interleaving shown in 3 colors.

A conceptual representation of the processing of Symbols in the Flit Mode for the 64.0 GT/s Data Rate on the Transmit and the Receive side is demonstrated in § Figure 4-21 , § Figure 4-22 , and § Figure 4-25 . A conceptual representation of the Flit Mode and Non-Flit Mode processing with 8b/10b encoding for 2.5 GT/s and 5.0 GT/s as well as with 128b/130b encoding for 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s for the Transmit and Receive side is shown in and respectively.

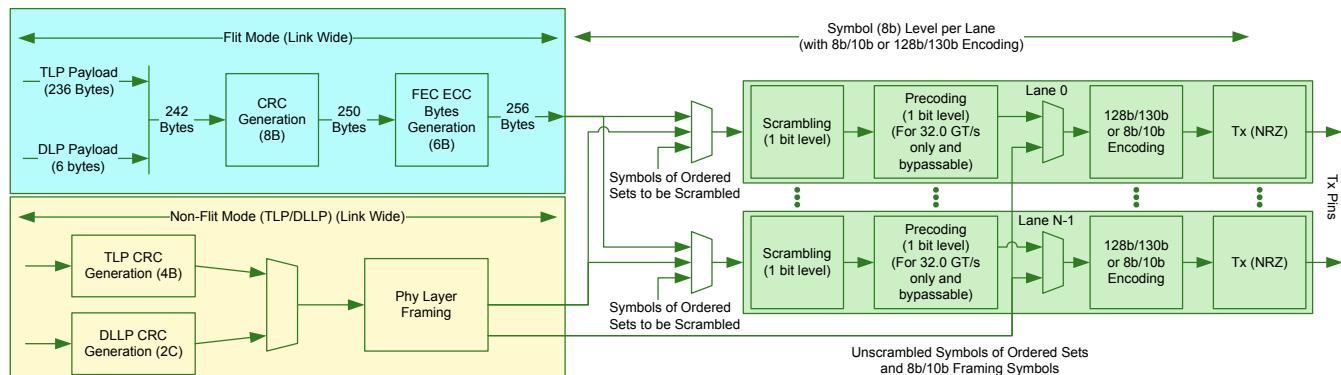


Figure 4-27 Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Transmit side §

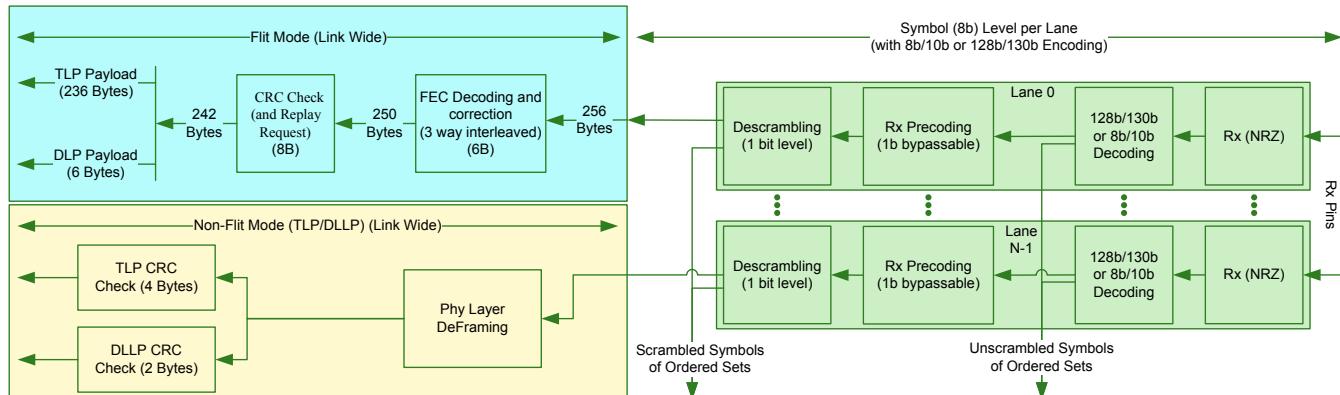


Figure 4-28 Flit Mode and Non-Flit Mode processing with 8b/10b and 128b/130b encoding on the Receive side §

4.2.3.4 Bytes in Flit Layout §

4.2.3.4.1 TLP Bytes in Flit §

The TLP Bytes in the flit carry Transaction Layer TLPs. Since the Flit Mode does not support STP tokens, these TLP bytes must be populated by the Transaction Layer, irrespective of whether it has a TLP to send or not. A TLP may span across multiple flits depending on its length and placement. The following rules must be observed:

- When the transaction layer does not have a TLP to send, it sends a NOP TLP (1DW). Once a NOP TLP is scheduled to be sent, NOP TLPs must be scheduled till the next 4DW aligned boundary within the Flit or the Flit boundary, whichever is earlier.
- NOP TLPs do not consume any credits.
- The TLPs in Flit Mode have information in predetermined position to determine the length of the TLPs, including the case where TLP prefix is used. See § Chapter 2. for details.
- The following rules apply for the non- NOP TLPs in bytes 0 through 127 of the Flit (i.e., the TLP bytes within the first Flit half) and bytes 128 through 235 of the Flit (i.e., the TLP bytes within the second Flit half):
 - No more than 8 TLPs, including partial TLPs. Receivers are permitted to check this rule.
 - If checked, this is logged as a Data Link Protocol Error in the receiving Port (see § Section 6.2.).
 - If a TLP at the end of a Flit gets poisoned or nullified through Flit_Status, and if that TLP extends to subsequent Flits, each of those Flits must also be poisoned or nullified through Flit_Status so that the poisoned or nullified Flit_Status is set in the Flit where the TLP ends. A Receiver is permitted to only look at the Flit_Status field in the Flit where the TLP ends.
 - A TLP that gets nullified through the Flit_Status field is ignored by the Receiver if the Flit is valid, but the credits must be released.
 - A TLP that gets poisoned or nullified through the Flit_Status field must be succeeded by only NOP TLPs through the end of the Flit.

IMPLEMENTATION NOTE: HANDLING TLPS SPANNING MULTIPLE FLITS §

When a TLP spans multiple Flits, it may be interrupted by SKPs, NOP Flits, transitions through Recovery and/or other unavoidable scenarios. Transmitters should attempt to avoid this interruption whenever possible for optimal bandwidth and latency and Receivers must be able to deal with scenarios where the TLP may be interrupted.

Base 6.4 vs Base 6.3

Example: § Table 4-15 shows an example of TLP placement in the Flit Mode for a x16 Link. For narrower Link widths, similar arrangements will exist subject to the rules mentioned above.

- This flit starts with a continuation of the remaining 2 DWs from TLP 19 (the 3rd DW of Hdr and 1 DW of Data).
- TLP 20 (4DW Hdr and 1DW Data) immediately starts in Lane 8 and ends in Lane 11 in the following Symbol.
- Since the Transmitter has nothing to send, it sends NOP sending till the 4DW aligned boundary, which is in Lane 15.
- TLPs 22, and 23 are scheduled without any intervening NOP.
- After TLP 23, the Transmitter did not have anything to send and sends 7 NOPs, aligned to a 4DW boundary till TLP 24 is ready, which continues till the flit boundary for TLP Bytes.
- After that we have 6 Bytes of DLP, 8 Bytes of CRC covering the 236 Bytes of TLP and 6 Bytes of DLP in the flit.
- Then we have the 3 sets of interleaved ECC, 2 bytes each, covering the entire 256B flit.

It should be noted that every Byte of each TLP as well the DLP is covered by one of the 3 ECC groups and each TLP/DLP is a member of all 3 ECC groups, and as indicated in the color code.

Table 4-15 Example TLP Placement in Flit Mode on a x16 Link §

| | | Lane | | | | | | | | | | | | | | | |
|-----------------------|---------------------|----------------|----|----|----|-----|-----|-----|--------------|-----|-----|-----|-----|-----|-----|-----|-----|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 Symbol times | TLP Bytes [0...223] | H2 | H2 | H2 | H2 | D0 | D0 | D0 | D0 | H0 | H0 | H0 | H0 | H1 | H1 | H1 | H1 |
| | | ... TLP 19 → | | | | | | | ← TLP 20 ... | | | | | | | | |
| | | H2 | H2 | H2 | H2 | H3 | H3 | H3 | H3 | D0 |
| | | ... TLP 20 → | | | | | | | NOP | | | | | | | | |
| | | H0 | H0 | H0 | H0 | H1 | H1 | H1 | H1 | H2 | H2 | H2 | H2 | H3 | H3 | H3 | H3 |
| | | ← TLP 21 ... | | | | | | | ... | | | | | | | | |
| | | D0 | D0 | D0 | D0 | D1 | D1 | D1 | D1 | D0 |
| | | ... TLP 21 → | | | | | | | NOP | | | | | | | | |
| | | H0 | H0 | H0 | H0 | H1 | H1 | H1 | H1 | H2 | H2 | H2 | H2 | H0 | H0 | H0 | H0 |
| | | ← TLP 22 → | | | | | | | ← TLP 23 ... | | | | | | | | |
| | | H1 | H1 | H1 | H1 | H2 | H2 | H2 | H2 | H3 | H3 | H3 | H3 | D0 | D0 | D0 | D0 |
| | | ... TLP 23 ... | | | | | | | ... | | | | | | | | |
| | | D1 | D1 | D1 | D1 | D2 | D2 | D2 | D2 | D3 | D3 | D3 | D3 | D4 | D4 | D4 | D4 |
| | | ... TLP 23 ... | | | | | | | ... | | | | | | | | |
| | | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 | D5 |
| | | ... TLP 23 → | | | | | | | NOP | | | | | | | | |
| | | NOP | | | | | | | | | | | | | | | |
| | | H0 | H0 | H0 | H0 | H1 | H1 | H1 | H1 | H2 | H2 | H2 | H2 | D0 | D0 | D0 | D0 |
| | | ← TLP 24 ... | | | | | | | ... | | | | | | | | |
| | | D1 | D1 | D1 | D1 | D2 | D2 | D2 | D2 | D3 | D3 | D3 | D3 | D4 | D4 | D4 | D4 |
| | | ... TLP 24 ... | | | | | | | ... | | | | | | | | |
| | | D5 | D5 | D5 | D5 | D6 | D6 | D6 | D6 | D7 | D7 | D7 | D7 | D7 | D8 | D8 | D8 |
| | | ... TLP 24 ... | | | | | | | ... | | | | | | | | |
| | | D9 | D9 | D9 | D9 | D10 | D10 | D10 | D10 | D11 | D11 | D11 | D11 | D12 | D12 | D12 | D12 |

| Description | Lane | | | | | | | | | | | | | | | |
|---------------------|----------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-------------|-------------|-------------|-------------|-------------|-------------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | ... TLP 24 ... | | | | | | | | | | | | | | | |
| | D13 | D13 | D13 | D13 | D14 | D14 | D14 | D14 | D15 | D15 | D15 | D15 | D16 | D16 | D16 | D16 |
| | ... TLP 24 ... | | | | | | | | | | | | | | | |
| TLP, DLP | D17 | D17 | D17 | D17 | D18 | D18 | D18 | D18 | D19 | D19 | D19 | DLP 0 | DLP 1 | DLP 2 | DLP 3 | ← DLP ... |
| DLP, CRC, ECC | DLP 4 | DLP 5 | CRC 0 | CRC 1 | CRC 2 | CRC 3 | CRC 4 | CRC 5 | CRC 6 | CRC 7 | ECC1 [0] | ECC2 [0] | ECC0 [0] | ECC1 [1] | ECC2 [1] | ECC0 [1] |
| | ... DLP → | | | | ← CRC → | | | | ← ECC → | | | | | | | |

4.2.3.4.2 DLP Bytes in Flit §

In every flit, 6 bytes are allotted to carry the information carried by the Data Link Layer Packets (DLLPs) described in § Chapter 3.. While the DLP bytes are designed to reuse the DLLP mechanism and formats for the most part, some optimizations have been made to effectively carry the Ack/Nak flit along with an optimized credit release (Optimized_Update_FC , defined below) or a traditional DLLP. An encoding has been added in the DLP to provide UpdateFC credits for PRH, PRD, and NPRH within one 32 bit payload of DLP. The following table summarizes the DLP encodings.

Three Flit Types have been defined in § Table 4-16 , along with their intended usages. They all have the same CRC and FEC mechanism. The distinction is made for ease of reference based on the TLP and DLP Bytes they carry. IDLE Flits are only sent after the Control SKP Ordered Set following the SDS Ordered Set Sequence with 128b/130b and 1b/1b encoding. For 8b/10b encoding, an IDLE Flit will be sent at the conclusion of the TS2 Ordered Set (or after a SKP OS following the last TS2 Ordered Set, depending on the SKP insertion interval). With 8b/10b encoding, an Ordered Set vs. a Flit is distinguished the lack of a COM at the end of a SKP OS . IDLE Flits continue to be sent until a Flit other than an IDLE Flit must be sent. Devices that advertise infinite credits across all FC/VC are permitted to send NOP DLLP if there is no DLLP to be sent in DLP 2, 3, 4, and 5.

All received Flits meet one or more of the following definitions:

- A **valid Flit** is one where all of the following are true:
 - CRC check passes after performing FEC correction (if needed)
 - No ECC group of the FEC is reporting an uncorrectable error (by ECC), as described in § Section 4.2.3.4.5
 - The Flit Usage field does not have a Reserved encoding (see § Figure 4-30).
 - The Flit_Status field does not have a Reserved encoding (see § Table 4-19).
- An FEC-correctable Flit is a **valid Flit** that passes the CRC check after performing the FEC correction.
- An FEC uncorrectable error is an error in a Flit that is detected by the CRC after FEC correction or ‘uncorrectable error’ reported by an ECC group in the FEC which may result in a NAK and a replay.
- An IDLE Flit is a Flit with the Flit Usage 00b, Sequence Number 0 and Replay Command 00b as described in § Table 4-16 . Implementations are permitted but not recommended to check additional fields.
- An invalid Flit is a Flit that is not a **valid Flit**.
- A valid non- IDLE Flit is a **valid Flit** that is either a NOP Flit or a Payload Flit . All subsequent Flits after the first valid non- **IDLE Flit** are non- **IDLE Flits** while the Link is in L0 .



Figure 4-29 DLP Byte to Bit Number Assignment §

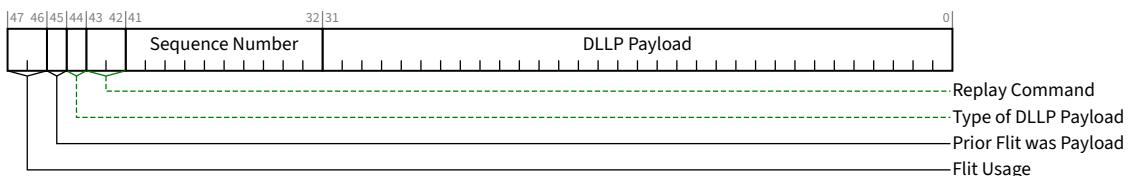


Figure 4-30 DLP Bit usage §

Table 4-16 Flit Types §

| Flit Type | TLP Bytes | DLP Bytes 0,1 | DLP Bytes 2..5 |
|---------------------|--|---|------------------------------|
| IDLE Flit | Transmitters must place 00h in all 236 bytes. | Transmitters must place 00h in both bytes. | NOP2 DLLP (see § Figure 3-8) |
| NOP Flit | NOP Flit Payload (see § Section 4.2.3.4.3) | Flit Usage = 00b Flit Sequence Number ⁶⁷ = NEXT_TX_FLIT_SEQ_NUM - 1 or, if REPLAY_IN_PROGRESS is 1b, the Sequence Number of the most recent Payload Flit that was sent Other Fields = Any valid encoding | Any valid encoding |
| Payload Flit | TLP content, packed per rules described in § Section 4.2.3.4.1 . A portion of at least one non- NOP TLP in the 236 Bytes | Flit Usage = 01b Flit Sequence Number != 00b if Replay Command is 00b Other Fields = Any valid encoding | Any valid encoding |

Table 4-17 DLP Bytes in the Flit §

| Field | Location | Encoding | | |
|--------------------------------|-------------------|---------------|--|--|
| Flit Usage | DLP0: Bits 7:6 | 00b | IDLE Flit or NOP Flit | |
| | | 01b | Payload Flit | |
| | | Others | Reserved | |
| Prior Flit was Payload | DLP0: Bit 5 | 0b | Prior flit was a NOP Flit or IDLE Flit | |
| | | 1b | Prior flit was a Payload Flit | |
| Type of DLLP Payload | DLP0: Bit 4 | 0b | DLLP Payload in DLP 2..5 | |
| | | 1b | Optimized_Update_FC or Flit_Marker in DLP 2..5 | |
| Replay Command [1:0] | DLP0: Bits 3:2 | 00b | The Flit Sequence Number included in the Flit is an Explicit Flit Sequence Number of the transmitted Flit. | |

67. A NOP Flit does not consume a Flit Sequence Number .

Base 6.4 vs Base 6.3

| Field | Location | Encoding |
|--------------------------------------|--|--|
| | | 01b Ack of Flit Sequence Number from Receiver. The included Flit Sequence Number indicates the Flit Sequence Number of the last valid Flit received. 10b Nak requesting a Replay of all unacknowledged Flits. The included Flit Sequence Number indicates the Flit Sequence Number of the last valid Flit received. 11b Nak requesting a Replay of a single Flit (Flit Sequence Number + 1). The included Flit Sequence Number indicates the Flit Sequence Number of the last valid Flit received. |
| Flit Sequence Number [9:0] | { DLP0: Bits 1:0, DLP1: Bits 7:0 } | 10-bit sequence number applied to Flits. See Flit Sequence Number and Retry Mechanism . DLP0[1:0] contains Flit Sequence Number [9:8] and DLP1[7:0] has Flit Sequence Number [7:0]. |
| DLLP Payload | { DLP2, DLP3, DLP4, DLP5 } | Type of DLLP Payload is 0b Regular DLLP Payload (see § Chapter 3.) |
| Optimized_Update_FC | Bits 31:0 DLP2 is 31:24 DLP3 is 23:16 DLP4 is 15:8 DLP5 is 7:0 | Type of DLLP Payload is 1b and bit 31 = 0b (see § Table 4-18) |
| Flit_Marker | | Type of DLLP Payload is 1b and bit 31 = 1b (see § Table 4-19) |

In the Table above, a Transmitter must not use the Reserved encodings.

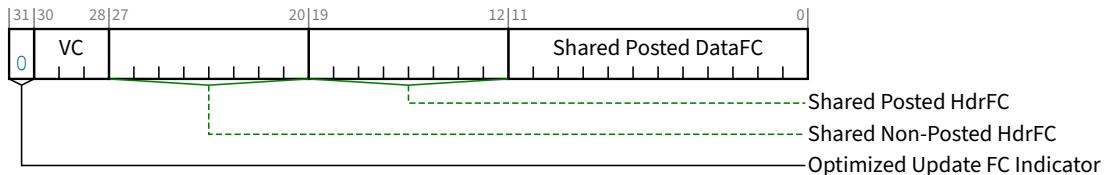


Figure 4-31 Optimized_Update_FC §

Table 4-18 Optimized_Update_FC §

| Bit Location | Description |
|--------------|--|
| 31 | Optimized Update FC Indicator Must be "0" |
| 30:28 | VC |
| 27:20 | Shared Non-Posted HdrFC – Shared credits on this VC |
| 19:12 | Shared Posted HdrFC – Shared credits on this VC |
| 11:0 | Shared Posted DataFC – Shared credits on this VC |

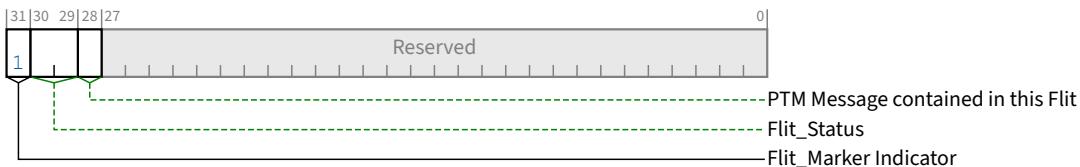


Figure 4-32 Flit_Marker §

Table 4-19 Flit_Marker §

| Bit Location | Description |
|--------------|---|
| 31 | Flit_Marker Indicator Must be "1" |
| 30:29 | <p>Flit_Status - Indicates the validity of the Last TLP in this Flit. Encoding is:</p> <ul style="list-style-type: none"> 00b No special information - No TLPs that end in this Flit are Nullified or Poisoned. TLPs that starts in this Flit and ends in a subsequent Flit are marked in that Flit. 01b Last TLP Nullified - Last TLP ending in current Flit is Nullified This mechanism replaces the EDB Mechanism in Non-Flit Mode framing. 10b Last TLP Poisoned - Last TLP ending in the current Flit is Poisoned See § Section 2.7.2 regarding how this mechanism relates to the Transaction Layer EP bit mechanism. 11b Reserved |
| 28 | <p>PTM Message contained in this Flit - In Flit Mode, timestamps for Precision Time Measurement messages are measured at the Flit level rather than at the TLP level. This bit indicates that this Flit contains the last symbol of a non-nullified PTM Message.</p> <p>Nullified PTM Messages do not set this bit.</p> <p>Poisoned PTM Messages are not permitted.</p> |

For the DLLP Payload in the above table, the Optimized_Update_FC is used to transmit performance critical NPRH, PRH, and PRD credits in 4 Bytes. For debug as well as ease of use by debug tools such as Logic Analyzers, devices must send at least one DLLP every 10 µs with an UpdateFC DLLP per VC with the scaled credit information. It is strongly recommended that a Transmitter cycles through the VCs with finite non-0 credits in the Optimized_Update_FC as long as there is a credit to be released in the corresponding VC.

Receivers are permitted to check for the following Flit errors:

- Reserved value of Flit Usage (see § Figure 4-30).
- Reserved encoding of Flit_Status field in a Flit_Marker (see § Table 4-19).
- Invalid sequence number (see § Section 4.2.3.4.2.1.4).
- Violations of the TLP packing rules as defined in § Section 4.2.3.4.1

If checked, these Flit errors are logged as a Data Link Protocol Error in the receiving Port (see § Section 6.2).

Three NOP Flit types have been defined in , along with their intended usages. All types defined below are considered NOP Flits and must follow all NOP Flit rules outlined in this specification. Refer to for the usage model of NOP.Debug and NOP.Vendor Flit types. The NOP Flit types defined below may rely on information provided by upper layers, but the transmission is done solely by the Physical Layer independently of the upper layers. Transmission may begin as soon as

Base 6.4 vs Base 6.3

the Link enters L0. For NOP.Empty and NOP.Debug Flits, no credit checks are required for transmission. For NOP.Vendor Flits, implementation of any credit mechanism and associated checking for the credits is implementation specific. A receiver that comprehends non-zero encodings must silently drop any NOP.Debug or NOP.Vendor Flits if it does not support processing them. Receiver behavior for designs that do support processing NOP.Debug or NOP.Vendor Flits is implementation specific but transmitting/receiving NOP.Debug and NOP.Vendor Flits must not affect the link state. NOP Flit Types NOP Flit Type Usage NOP.Empty Empty payload (see) NOP.Debug Deliver debug information (see) NOP.Vendor Deliver vendor defined information (see) All NOP Flit TLPs use a common definition of the first DW of the NOP Flit Payload. Subsequent content depends on the value of NOP Flit Type (see). { "debug": false, "preClass": "hide", "width": 128, "cellwidth": 15, "isMessage": true, "defaultUnused": "Reserved", "fields": [{ "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "NOP Flit Type", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 3, "msbit": 7, "name": "NOP Stream ID", "attr": "ro" }, { "lsbyte": 15, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "[232 bytes of content vary based on NOP Flit Type value]", "attr": "ro" }] } NOP Flit Common Header NOP Flit Common Header Fields Field Location Definition NOP Flit Type Byte 0: Bits 7:4 0h NOP.Empty Flit 1h NOP.Debug Flit 2h to Eh Reserved. Receiver must treat as a NOP.Empty Flit Fh NOP.Vendor Flit NOP Flit Counter [Byte 0: Bits 3:0, Byte 1: Bits 7:0] NOP.Empty Flit Transmitter populates with 000h, Receiver is permitted to ignore NOP.Debug Flit and NOP.Vendor Flit Incrementing per NOP Flit Type counter of NOP.Debug and NOP.Vendor Flit types Reserved Byte 2: Bits 7:0 Reserved NOP Stream ID Byte 3: Bits 7:0 NOP.Empty Flit Transmitter populates with 00h, Receiver is permitted to ignore NOP.Debug Flit and NOP.Vendor Flit Original source identifier of the NOP Flit A non-empty NOP Flit is a NOP Flit with a non-zero NOP Flit Type encoding. A NOP Stream is a sequence of NOP Flits that are sourced by a Transmitter. The NOP Stream ID field in the NOP Flit provides a mechanism to identify the original Transmitter of the NOP Flit. The NOP Flit Extended Capability structure (see) provides software control (via the NOP Stream ID Start and Number of NOP Streams fields) over the range of values that a Transmitter may use. The usage details of the programmed range are implementation specific. The combination of NOP Flit Type value and NOP Stream ID value uniquely identifies a NOP Stream and its value must be unique across all entities transmitting NOP Flits over a given Link. NOP.Empty Flits must always transmit a NOP Stream ID value of 00h. The NOP Flit Counter field in the NOP Flit allows dropped or missing non-empty NOP Flits to be detected. Every NOP Stream must have an independent counter. Each NOP Stream must start with a Nop Flit Counter value of 000h and every subsequent non-empty NOP Flit of the NOP Stream that is transmitted must increment the counter by one. The NOP Flit Counter must roll over after FF Eh. A NOP Flit Counter value of FFFh indicates an error or unexpected behavior occurred. This provides an indication that an unknown number of Flits of a NOP Stream were lost. This indication is only a hint since the Flit containing this indication could itself be lost. When a NOP Stream is disabled, its NOP Flit Counter must reset to 000h. NOP.Empty Flits must always transmit a NOP Flit Counter value of 000h. Note: Receivers may forward received NOP Flits with a NOP Stream ID other than FFh to a different Link. In this scenario, the forwarded NOP Flit must preserve all fields as received, including the NOP Flit Type, the NOP Flit Counter and the NOP Stream ID. The forwarding Receiver is permitted to overwrite the NOP Flit Counter to FFFh in certain situations. The forwarding mechanism is implementation specific on a best-effort basis, with no guarantee of delivery. After the first DW, the remaining TLP Bytes carry the NOP Flit Payload, whose definition varies depending on the NOP Flit Type. See for NOP Flit Payload definitions.↓

4.2.3.4.2.1 Flit Sequence Number and Retry Mechanism §

Term Definitions

Explicit Sequence Number Flit

A Payload Flit or NOP Flit with Replay Command 00b.

Ack Flit

A Flit with Replay Command 01b.

Standard Nak Flit

A Flit with Replay Command 10b.

Selective Nak Flit

A Flit with Replay Command 11b.

Nak Flit

A Flit with either Replay Command 10b or 11b.

Standard Nak

A Nak that requests a Replay of all unacknowledged Flits.

Selective Nak

A Nak that requests a Replay of a specific Flit.

Standard Replay

A Replay of all unacknowledged Flits in the TX Retry Buffer.

Selective Replay

A Replay of a specific Flit from the TX Retry Buffer.

TX Retry Buffer

The buffer which stores information for transmitted Flits until the Flit has been acknowledged by the Link partner.⁶⁸

RX Retry Buffer

The buffer which stores information for received Flits until the Flit has been consumed by the Receiver.⁶⁹

Nak Ignore Window

A time window in which received Nak Flits are ignored for a specific Flit Sequence Number.

- The Nak Ignore Window is started when a Payload Explicit Sequence Number Flit is sent with Flit Sequence Number = NAK_IGNORE_FLIT_SEQ_NUM + 1 and NAK_IGNORE_FLIT_SEQ_NUM
I= 0.⁷⁰

Flags and Counters**NEXT_TX_FLIT_SEQ_NUM**

10-bit unsigned counter that tracks the Flit Sequence Number for transmitted Flits.

- Set to 001h in DL_Inactive state.
- Behavior defined in NEXT_TX_FLIT_SEQ_NUM Rules.
- Usage defined in § Section 4.2.3.4.2.1.2, § Section 4.2.3.4.2.1.3, § Section 4.2.3.4.2.1.4, and § Section 4.2.3.4.2.1.6.

TX_ACKNAK_FLIT_SEQ_NUM

Stores the 10-bit Sequence Number to be transmitted in the next Ack Flit or Nak Flit.

- Set to 1023 in DL_Inactive state.
- Usage defined in § Section 4.2.3.4.2.1.2, § Section 4.2.3.4.2.1.3, and § Section 4.2.3.4.2.1.5.

CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS

2-bit unsigned counter which tracks how many consecutive Explicit Sequence Number Flits have been sent by the Transmitter.

- Set to 00b IDLE Flit Handshake Phase.
- Behavior defined in CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS Rules.
- Usage defined in § Section 4.2.3.4.2.1.2 and § Section 4.2.3.4.2.1.3.

CONSECUTIVE_TX_NAK_FLITS

3-bit unsigned counter which tracks how many consecutive Nak Flits have been sent by the Transmitter.

- Set to 000b in the IDLE Flit Handshake Phase.
- This counter saturates at 111b and does not roll over to 000b.

68. The Link Speed and Link Width that a Device supports should be taken into consideration when sizing the TX Retry Buffer.

69. Expected Nak latency should be considered when sizing the RX Retry Buffer to avoid overflow when using the Selective Replay mechanism.

70. The Nak Ignore Window is used in the two specific scenarios: 1) A Replay is started due to a received Nak Flit. 2) A Payload Explicit Sequence Number Flit with Flit Sequence Number = NEXT_TX_FLIT_SEQ_NUM is sent after having received a Nak with Flit Sequence Number = NEXT_TX_FLIT_SEQ_NUM - 1.

- Behavior defined in CONSECUTIVE_TX_NAK_FLITS Rules .
- Usage defined in § Section 4.2.3.4.2.1.3 and § Section 4.2.3.4.2.1.7 .

NEXT_EXPECTED_RX_FLIT_SEQ_NUM

10-bit unsigned counter that stores the expected Flit Sequence Number of the next valid non-IDLE non-duplicate Flit to be received.

- Set to 001h in DL_Inactive state.
- Behavior and usage defined in § Section 4.2.3.4.2.1.5 .

IMPLICIT_RX_FLIT_SEQ_NUM

10-bit unsigned counter that tracks the implicit Flit Sequence Number of received Flits.

- Set to 000h in DL_Inactive state.
- Behavior defined in IMPLICIT_RX_FLIT_SEQ_NUM Rules .
- Usage defined in § Section 4.2.3.4.2.1.5 .

ACKD_FLIT_SEQ_NUM

Stores the 10-bit Flit Sequence Number received in the most recently processed valid Ack Flit or Nak Flit .

- Set to 3FFh in DL_Inactive state.
- Behavior defined in § Section 4.2.3.4.2.1.4 .
- Usage defined in § Section 4.2.3.4.2.1.4 and § Section 4.2.3.4.2.1.6 .

NON_IDLE_EXPLICIT_SEQ_NUM_FLIT_RCVD

Flag to indicate that a Non-IDLE Explicit Sequence Number Flit has been received since the last entry to IDLE Flit Handshake Phase .

- Set to 1 when first Non-IDLE Explicit Sequence Number Flit is received after last entry to IDLE Flit Handshake Phase .
 - This must be updated before evaluating Ack, Nak, and Discard Rules in § Section 4.2.3.4.2.1.5 .
- Cleared on entry to IDLE Flit Handshake Phase .
- Usage defined in IMPLICIT_RX_FLIT_SEQ_NUM Rules .

REPLAY_SCHEDULED

Flag to indicate that a Flit has been scheduled to be replayed from the TX Retry Buffer .

- Cleared when in DL_Inactive state.
- Behavior defined in § Section 4.2.3.4.2.1.6 .
- Usage defined in § Section 4.2.3.4.2.1.2 , § Section 4.2.3.4.2.1.6 , and § Section 4.2.3.4.2.1.7 .

REPLAY_SCHEDULED_TYPE

Indicates whether a Standard Replay or a Selective Replay has been scheduled.

- If REPLAY_SCHEDULED is 1b:
 - REPLAY_SCHEDULED_TYPE 0b indicates a Standard Replay .
 - REPLAY_SCHEDULED_TYPE of 0b will be referred to as **STANDARD_REPLAY** .
 - REPLAY_SCHEDULED_TYPE 1b indicates a Selective Replay .
 - REPLAY_SCHEDULED_TYPE of 1b will be referred to as **SELECTIVE_REPLAY** .
- Behavior defined in § Section 4.2.3.4.2.1.6 .
- Usage defined in § Section 4.2.3.4.2.1.6 and § Section 4.2.3.4.2.1.7 .

REPLAY_IN_PROGRESS

Flag to indicate that the Transmitter is sending Flits from the TX Retry Buffer .

- Cleared when in DL_Inactive state.
- Behavior defined in REPLAY_IN_PROGRESS Rules .
- Usage defined in § Section 4.2.3.4.2.1.2 , § Section 4.2.3.4.2.1.6 , and § Section 4.2.3.4.2.1.7 .

TX_REPLAY_FLIT_SEQ_NUM

Stores the 10-bit Flit Sequence Number of the first Flit to be replayed from the TX Retry Buffer .

- Set to 000h in DL_Inactive state.
- Behavior defined in § Section 4.2.3.4.2.1.6 .
- Usage defined in § Section 4.2.3.4.2.1.4 , § Section 4.2.3.4.2.1.6 , and § Section 4.2.3.4.2.1.7 .

FLIT_REPLAY_NUM

3-bit unsigned counter that tracks the number of times that a Replay has been initiated without making forward progress.

- Set to 000b in DL_Inactive state.
- Behavior defined in § Section 4.2.3.4.2.1.4 and § Section 4.2.3.4.2.1.7 .
- Usage defined in § Section 4.2.3.4.2.1.7 .

REPLAY_TIMEOUT_FLIT_COUNT

11-bit unsigned counter that counts the number of Payload Flits and NOP Flits sent since the last Ack of an outstanding Flit.

- Set to 000h in DL_Inactive state.
- This counter saturates at 7FFh and does not roll over to 000h.
- Behavior defined in REPLAY_TIMEOUT_FLIT_COUNT Rules
- Usage defined in § Section 4.2.3.4.2.1.6 .

NAK_SCHEDULED

Flag to indicate that a Nak Flit has been scheduled for transmission.

- Cleared when in DL_Inactive state.
- Behavior defined in § Section 4.2.3.4.2.1.5 .
- Usage defined in § Section 4.2.3.4.2.1.2 , § Section 4.2.3.4.2.1.3 , and § Section 4.2.3.4.2.1.5 .

NAK_SCHEDULED_TYPE

Indicates whether a Standard Nak Flit or a Selective Nak Flit should be sent.

- Set to 0b when in DL_Inactive state.
- If NAK_SCHEDULED is 1b,
 - NAK_SCHEDULED_TYPE 0b indicates a Standard Nak Flit .
 - NAK_SCHEDULED_TYPE of 0b will be referred to as **STANDARD_NAK** .
 - NAK_SCHEDULED_TYPE 1b indicates a Selective Nak Flit .
 - NAK_SCHEDULED_TYPE of 1b will be referred to as **SELECTIVE_NAK** .
- Behavior defined in § Section 4.2.3.4.2.1.5 .
- Usage defined in § Section 4.2.3.4.2.1.2 , § Section 4.2.3.4.2.1.3 , and § Section 4.2.3.4.2.1.5 .

NAK_WITHDRAWAL_ALLOWED

Flag that indicates an invalid Flit was received but should not be Nak'd if the subsequent received Flit has Prior Flit was Payload set to 0b (indicating that the invalid Flit was a NOP Flit).

- Cleared when in DL_Inactive state.
- Behavior and usage defined in § Section 4.2.3.4.2.1.5 .

NAK_IGNORE_FLIT_SEQ_NUM

Stores the 10-bit Flit Sequence Number to be ignored during the Nak Ignore Window .

- Set to 000h when in DL_Inactive state.
- Behavior defined in § Section 4.2.3.4.2.1.4 and § Section 4.2.3.4.2.1.6
- Usage defined in § Section 4.2.3.4.2.1.6 .

NEXT_RX_FLIT_SEQ_NUM_TO_STORE

Stores the 10-bit Flit Sequence Number of the next Flit to be stored in the RX Retry Buffer .

- Cleared when in DL_Inactive state.
- Behavior and usage defined in § Section 4.2.3.4.2.1.5 .

RX_RETRY_BUFFER_OVERFLOW

Flag to indicate that a Flit was unable to be stored in the RX Retry Buffer due to the buffer becoming full before a Selective Replay was received for an outstanding Selective Nak .

- Cleared when in DL_Inactive state.
- Behavior and usage defined in § Section 4.2.3.4.2.1.5 .

RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM

Stores the 10-bit Flit Sequence Number of the last Flit stored in the RX Retry Buffer if the RX Retry Buffer overflows.

- Cleared when in DL_Inactive state.
- Behavior and usage defined in § Section 4.2.3.4.2.1.5 .

MAX_UNACKNOWLEDGED_FLITS

Maximum number of unacknowledged Flits outstanding.

- Set to the lesser of:
 - Number of Flits that can be stored in the TX Retry Buffer .
 - 511 Flits
- Usage defined in § Section 4.2.3.4.2.1.4 , § Section 4.2.3.4.2.1.5 , and § Section 4.2.3.4.2.1.6 .

General Rules**Flit Sequence Number Rules:**

- The valid range of Flit Sequence Numbers for Payload Flits is 1-1023.
- Flit Sequence Number 0 is only used for IDLE Flits .
- NOP Flits do not consume a Flit Sequence Number .
 - A NOP Flit that is also an Explicit Sequence Number Flit sets Flit Sequence Number to NEXT_TX_FLIT_SEQ_NUM - 1.
 - If no Payload Flits have been transmitted, Flit Sequence Number 1023 is used.
 - If REPLAY_IN_PROGRESS is 1b, the transmitter is strongly recommended to set the Explicit Sequence Number to the Sequence Number of the most recent Payload Flit that was sent.
- All Flit Sequence Number counters will roll-over from 1023 to 1.
- Whenever text in this section refers to adding to, subtracting from, incrementing, or decrementing a sequence number, it is implied that sequence number rollover rules stated above will be followed.
- Two transmitted or received Flits are considered consecutive, even if they are separated by a SKP OS.
- The Nak Ignore Window should be the lesser of:
 - Measured Ack/Nak Latency of the Link + 1 maximum sized SKP Ordered Set + 2 Flits

Base 6.4 vs Base 6.3

- ~~Either the value found in § Table 4-20 (strongly recommended) or 300 ns⁷¹~~

Errata: Base 6.3
B837△▷

Table 4-20 ~~Nak Ignore Window~~⁷² §

Errata: Base 6.3
B837△▷

| ↑↓2.5 GT/s↑ | ↑↓5.0 GT/s↑ | ↑↓8.0 GT/s↑ | ↑↓16.0 GT/s↑ | ↑↓32.0 GT/s↑ | ↑↓64.0 GT/s↑ |
|-------------|-------------|-------------|--------------|--------------|--------------|
| ↑↓2250 ns↑ | ↑↓1250 ns↑ | ↑↓750 ns↑ | ↑↓450 ns↑ | ↑↓350 ns↑ | ↑↓300 ns↑ |

↑↓↓↓

- ~~Unless the transmitter is sending an Ordered Set, it is strongly recommended that an Ack or Nak be transmitted within the following times of receiving the Flit on the Receiver pins. The following time shown in § Table 4-21. These times apply unless the transmitter is sending an Ordered Set. The measurement is from the first bit of the received Flit on the Receiver pins to the first bit of the associated Ack Flit or Nak Flit on the transmitter pins. § Section 4.2.3.4.2.1.5 describes when an Ack or Nak should be scheduled.~~⁷³

Errata: Base 6.3
B837△◀

Table 4-21 ~~Ack/Nak Latency~~⁷³ §

| | ↑↓2.5 GT/s↑ | ↑↓5.0 GT/s↑ | ↑↓8.0 GT/s↑ | ↑↓16.0 GT/s↑ | ↑↓32.0 GT/s↑ | ↑↓64.0 GT/s↑ |
|--------|-------------|-------------|-------------|--------------|--------------|--------------|
| ↑↓x16↑ | ↑↓112 ns↑ | ↑↓80 ns↑ | ↑↓64 ns↑ | ↑↓56 ns↑ | ↑↓52 ns↑ | 50 ns ↑↓x8:↓ |
| ↑↓x8↑ | ↑↓176 ns↑ | ↑↓112 ns↑ | ↑↓80 ns↑ | ↑↓64 ns↑ | ↑↓56 ns↑ | 52 ns ↑↓x4:↓ |
| ↑↓x4↑ | ↑↓304 ns↑ | ↑↓176 ns↑ | ↑↓112 ns↑ | ↑↓80 ns↑ | ↑↓64 ns↑ | 56 ns ↑↓x2:↓ |
| ↑↓x2↑ | ↑↓560 ns↑ | ↑↓304 ns↑ | ↑↓176 ns↑ | ↑↓112 ns↑ | ↑↓80 ns↑ | 64 ns ↑↓x1:↓ |
| ↑↓x1↑ | ↑↓1072 ns↑ | ↑↓560 ns↑ | ↑↓304 ns↑ | ↑↓176 ns↑ | ↑↓112 ns↑ | 80 ns |

Transmitter Rules

- The Transmitter must store the following information of transmitted Payload Flits in the TX Retry Buffer :
 - TLP Bytes associated with non- NOP TLPs
 - Flit_Status
 - Flit Sequence Number

MAX_UNACKNOWLEDGED_FLITS Outstanding:

- If $(\text{NEXT_TX_FLIT_SEQ_NUM} - \text{ACKD_FLIT_SEQ_NUM}) \bmod 1023 > \text{MAX_UNACKNOWLEDGED_FLITS}$:
 - The Transmitter must not accept any new TLPs from the Transaction Layer.
 - The Transmitter must send a NOP Flit if REPLAY_SCHEDULED is 0b and REPLAY_IN_PROGRESS is 0b.

REPLAY_IN_PROGRESS Rules :

- If REPLAY_IN_PROGRESS is 1b:
 - The Transmitter must not accept any new TLPs from the Transaction Layer.
 - REPLAY_IN_PROGRESS is cleared if either of the following conditions are true:

71. ~~Table 4-20 was added in [PCIe-6.4]. Earlier specifications defined a 300 ns minimum for all data rates. Using 300 ns at lower data rates can cause additional Replays due to the Nak Ignore Window being too short.~~

72. ~~These values are computed from: 200 ns + (x1 Flit Time * 2) rounded up to the nearest 50 ns~~

73. ~~The values in Table 4-21 are computed as: 48 ns + 1 Flit Time~~

- REPLAY_SCHEDULED_TYPE is STANDARD_REPLAY and all unacknowledged Flits in the TX Retry Buffer have been transmitted.
- REPLAY_SCHEDULED_TYPE is SELECTIVE_REPLAY and Flit with Flit Sequence Number equal to TX_REPLY_FLIT_SEQ_NUM has been transmitted.

CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS Rules

- When an Explicit Sequence Number Flit is transmitted:
 - CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is incremented.
- When a non- Explicit Sequence Number Flit is transmitted:
 - CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is set to 0b.
- Also see Replay Schedule Rule 0 .

REPLAY_TIMEOUT_FLIT_COUNT Rules :

- When a Payload Flit or a NOP Flit is transmitted and the TX Retry Buffer is not empty:
 - REPLAY_TIMEOUT_FLIT_COUNT is incremented.
- When an Ack or Nak is received that causes a Flit stored TX Retry Buffer to be purged:
 - REPLAY_TIMEOUT_FLIT_COUNT is set to 0b.
- Also see Replay Schedule Rule 0 .

NEXT_TX_FLIT_SEQ_NUM Rules

- After the Transmitter applies the Replay Command to a Payload Flit which is NOT being replayed from the TX Retry Buffer :
 - NEXT_TX_FLIT_SEQ_NUM is incremented.
- NEXT_TX_FLIT_SEQ_NUM does not change when sending any of the following:
 - An IDLE Flit
 - A NOP Flit
 - A Flit being replayed from the TX Retry Buffer

Receiver Rules

- If a Receiver has scheduled a Selective Nak Flit , it must continue to store non- NOP TLP bytes and Flit Sequence Number of subsequent Flits in the RX Retry Buffer .
 - After the Replay for a Selective Nak is received, received Flits must continue to be processed in order of Flit Sequence Number. The management of storage and processing of Flits received after the Selective Replay Flit is received is implementation specific
- See § Section 4.2.3.4.2.1.5 for Ack, Nak, and Discard rules.

IMPLICIT_RX_FLIT_SEQ_NUM Rules

- The value of IMPLICIT_RX_FLIT_SEQ_NUM must be updated before evaluating the rules in § Section 4.2.3.4.2.1.5
- IMPLICIT_RX_FLIT_SEQ_NUM does not change if ANY of the following are true:
 - A valid IDLE Flit is received.
 - A valid Explicit Sequence Number Flit is received with Flit Sequence Number 0.
 - Both of the following are true:
 - NON_IDLE_EXPLICIT_SEQ_NUM_FLIT_RCV is 0b
 - Either of the following are true:
 - An invalid Flit is received.

- A valid non- Explicit Sequence Number Flit is received.
- Both of the following are true:
 - A valid NOP Flit is received with a non-explicit Flit Sequence Number.
 - Either of the following are true:
 - NAK_WITHDRAWAL_ALLOWED is 0b
 - NAK_WITHDRAWAL_ALLOWED is 1b and Prior Flit was Payload is 1b
- All of the following are true:
 - A valid Payload Flit with a non-explicit Flit Sequence Number is received.
 - NAK_WITHDRAWAL_ALLOWED is 1b
 - Prior Flit was Payload is 0b
- IMPLICIT_RX_FLIT_SEQ_NUM is incremented if both of the following are true:
 - NON_IDLE_EXPLICIT_SEQ_NUM_FLIT_RCVD is 1b
 - Any of the following are true:
 - A valid Payload Flit with a non-explicit Flit Sequence Number is received and NAK_WITHDRAWAL_ALLOWED is 0b.
 - A valid Payload Flit with a non-explicit Flit Sequence Number is received with Prior Flit was Payload set to 1b and NAK_WITHDRAWAL_ALLOWED is 1b.
 - A invalid Flit is received.
- IMPLICIT_RX_FLIT_SEQ_NUM is set to N if both of the following are true:
 - A valid non-IDLE Explicit Sequence Number Flit is received with Flit Sequence Number N.
 - N != 0.
- IMPLICIT_RX_FLIT_SEQ_NUM is decremented if ALL of the following are true:
 - A valid NOP Flit with a non-explicit Flit Sequence Number is received.
 - NAK_WITHDRAWAL_ALLOWED is 1b.
 - Prior Flit was Payload is 0b.

4.2.3.4.2.1.1 IDLE Flit Handshake Phase §

General Rules

- A Port enters this phase when the LTSSM enters Configuration.Idle or Recovery.Idle states.

Transmitter Rules

- The Transmitter must send IDLE Flits as required by the LTSSM rules.
- The Transmitter moves to the Sequence Number Handshake Phase when the LTSSM enters the L0 state.

Receiver Rules

- All received Flits are discarded.
- The Receiver moves to the Sequence Number Handshake Phase when two consecutive valid IDLE Flits have been received.

4.2.3.4.2.1.2 Sequence Number Handshake Phase §

General Rules

- Devices are permitted to perform Data Link Feature Exchange and Flow Control Initialization in this phase if the Data Link Control and Management State Machine is in the DL_Feature or DL_Init states, respectively.
- The Port transitions to the Normal Flit Exchange Phase when both of the following are true:
 - The Port has transmitted 3 or more Ack Flits with non-zero Flit Sequence Numbers after having received one or more Explicit Sequence Number Flits with a non-zero Flit Sequence Number.
 - The Port has transmitted 9 or more Explicit Sequence Number Flits with a non-zero Flit Sequence Number.
- If the Port is in this phase after sending 512 Flits with 1b/1b encoding or 256 Flits with 8b/10b or 128b/130b encoding, the Port must enter Recovery.

Transmitter Rules

Flit Replay Command Rules

- The Transmitter alternates between sending three consecutive Explicit Sequence Number Flit and one Ack Flit or Nak Flit while in this phase.
 - The following conditions are listed in order of priority.
 - A Transmitter must send an Explicit Sequence Number Flit if the following is true:
 - CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS < 3.
 - A Transmitter must send a Standard Nak Flit if all of the following are true:
 - NAK_SCHEDULED is 1b.
 - NAK_SCHEDULED_TYPE is STANDARD_NAK.
 - The conditions for sending an Explicit Sequence Number Flit were not met.
 - A Transmitter must send a Selective Nak Flit if all of the following are true:
 - NAK_SCHEDULED is 1b.
 - NAK_SCHEDULED_TYPE is SELECTIVE_NAK.
 - The conditions for sending an Explicit Sequence Number Flit were not met.
 - The conditions for sending a Standard Nak Flit were not met.
 - A Transmitter must send an Ack Flit if all of the following are true:
 - The conditions for sending an Explicit Sequence Number Flit were not met.
 - The conditions for sending a Standard Nak Flit were not met.
 - The conditions for sending a Selective Nak Flit were not met.
- The Transmitter may send up to six consecutive Explicit Sequence Number Flits if the CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS counter is reset due to Replay Schedule Rule 0.

Flit Replay Transmit Rules

- The Transmitter must only begin replaying Flits from the TX Retry Buffer when sending an Explicit Sequence Number Flit and CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is 0b.
- If the Transmitter is replaying a Flit from the TX Retry Buffer, it must use the sequence number of the Flit from its TX Retry Buffer when sending an Explicit Sequence Number Flit.
- See § Section 4.2.3.4.2.1.7 for additional replay transmit rules.

Flit Sequence Number Rules

- When transmitting an Ack Flit or Nak Flit with a non-0 Flit Sequence Number:
 - The Flit Sequence Number is set to the value of TX_ACKNAK_FLIT_SEQ_NUM.
- When transmitting an Explicit Sequence Number Flit:
 - A NOP Flit must set Flit Sequence Number to NEXT_TX_FLIT_SEQ_NUM - 1.
 - If REPLAY_IN_PROGRESS is 1b, the transmitter is permitted to set the Explicit Sequence Number to the Sequence Number of the most recent Payload Flit that was sent.
 - A replayed Payload Flit must set Flit Sequence Number to the value stored for the Flit in the TX Retry Buffer.
 - A non-replayed Payload Flit must set Flit Sequence Number to NEXT_TX_FLIT_SEQ_NUM.

Receiver Rules

- Refer to § Section 4.2.3.4.2.1.4 for Ack and Nak processing rules.
- Refer to § Section 4.2.3.4.2.1.5 for Ack and Nak scheduling rules.
- Refer to § Section 4.2.3.4.2.1.6 for Replay scheduling rules.

4.2.3.4.2.1.3 Normal Flit Exchange Phase §

Transmitter Rules

- While in this Phase, Explicit Sequence Number Flits are only sent according to the Flit Replay Command Rules defined in this section.
- If a Nak Flit is sent in this phase:
 - A minimum of three consecutive Nak Flits must be sent unless any of the following are true:
 - CONSECUTIVE_TX_NAK_FLITS is reset due to Replay Schedule Rule 0.
 - The transmission of Flits is interrupted by an exit from the L0 state.
 - The NAK_SCHEDULED flag is cleared.
- If an Explicit Sequence Number Flit is sent in this phase:
 - Exactly three consecutive Explicit Sequence Number Flits must be sent unless either of the following are true:
 - CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is reset due to Replay Schedule Rule 0.
 - The transmission of Flits is interrupted by an exit from the L0 state.

Flit Replay Command Rules

- The following conditions are listed in order of priority.
- A Transmitter must send an Explicit Sequence Number Flit if all of the following are true:
 - Any of the following are true:
 - REPLAY_SCHEDULED is 1b.
 - REPLAY_IN_PROGRESS is 1b.
 - CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS > 0.
 - All of the following are true⁷⁴:
 - REPLAY_SCHEDULED is 0b.
 - REPLAY_IN_PROGRESS is 0b.

⁷⁴. This covers the case where a Nak is received with Flit Sequence Number = NEXT_TX_FLIT_SEQ_NUM - 1.

- NAK_IGNORE_FLIT_SEQ_NUM != 000h
- CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS < 3.
- Either of the following are true:
 - CONSECUTIVE_TX_NAK_FLITS equals 0.
 - CONSECUTIVE_TX_NAK_FLITS > 2.
- A Transmitter must send a Standard Nak Flit if all of the following are true:
 - NAK_SCHEDULED is 1b.
 - NAK_SCHEDULED_TYPE is STANDARD_NAK.
 - The conditions for sending an Explicit Sequence Number Flit were not met.
- A Transmitter must send a Selective Nak Flit if all of the following are true:
 - NAK_SCHEDULED is 1b.
 - NAK_SCHEDULED_TYPE is SELECTIVE_NAK.
 - The conditions for sending an Explicit Sequence Number Flit were not met.
 - The conditions for sending a Standard Nak Flit were not met.
- A Transmitter must send an Ack Flit if all of the following are true:
 - The conditions for sending an Explicit Sequence Number Flit were not met.
 - The conditions for sending a Standard Nak Flit were not met.
 - The conditions for sending a Selective Nak Flit were not met.
- The Transmitter may send up to six consecutive Explicit Sequence Number Flits if the CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS counter is reset due to Replay Schedule Rule 0.

Flit Replay Transmit Rules

- The Transmitter must only begin replaying Flits from the TX Retry Buffer when sending an Explicit Sequence Number Flit and CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is 0b.
- If the Transmitter is replaying a Flit from the TX Retry Buffer, it must use the sequence number of the Flit from its TX Retry Buffer when sending an Explicit Sequence Number Flit.
- See § Section 4.2.3.4.2.1.7 for additional replay transmit rules.

Flit Sequence Number Rules

- When transmitting an Ack Flit or Nak Flit:
 - The Flit Sequence Number is set to the value of TX_ACKNAK_FLIT_SEQ_NUM.
- When transmitting an Explicit Sequence Number Flit:
 - A NOP Flit must set Flit Sequence Number to NEXT_TX_FLIT_SEQ_NUM - 1.
 - If REPLAY_IN_PROGRESS is 1b, the transmitter is permitted to set the Explicit Sequence Number to the Sequence Number of the most recent Payload Flit that was sent.
 - A replayed Payload Flit must set Flit Sequence Number to the value stored for the Flit in the TX Retry Buffer.
 - A non-replayed Payload Flit must set Flit Sequence Number to NEXT_TX_FLIT_SEQ_NUM.

CONSECUTIVE_TX_NAK_FLITS Rules

- When a Nak Flit is transmitted:
 - CONSECUTIVE_TX_NAK_FLITS is incremented.
- When a non- Nak Flit is transmitted:

- CONSECUTIVE_TX_NAK_FLITS is set to 0b.
- Also see Replay Schedule Rule 0 .

Receiver Rules

- Refer to § Section 4.2.3.4.2.1.4 for Ack and Nak processing rules.
- Refer to § Section 4.2.3.4.2.1.5 for Ack and Nak scheduling rules.
- Refer to § Section 4.2.3.4.2.1.6 for Replay scheduling rules.

4.2.3.4.2.1.4 Received Ack and Nak Processing §

- A received Ack or Nak with Flit Sequence Number 0 must be ignored.
- Processing of received Ack or Nak Flits must not interrupt a replay in progress.
- Handling an invalid Flit Sequence Number in a received Ack Flit or Nak Flit :
 - If:
 - A valid Ack Flit or Nak Flit is received with Flit Sequence Number N where one or more of the following are true:
 - $((\text{NEXT_TX_FLIT_SEQ_NUM} - 1) - N) \bmod 1023 > \text{MAX_UNACKNOWLEDGED_FLITS}$
 - $(N - \text{ACKD_FLIT_SEQ_NUM}) \bmod 1023 > \text{MAX_UNACKNOWLEDGED_FLITS}$
 - Then:
 - The received Ack or Nak is ignored.
 - A Data Link Protocol Error is logged in the receiving Port (see § Section 6.2).
- If a Receiver receives a valid Ack Flit or Nak Flit with Flit Sequence Number N, it must:
 - Purge all Flits stored the TX Retry Buffer that are older than and including the Flit with Flit Sequence Number N.
 - If $(N - \text{ACKD_FLIT_SEQ_NUM}) \bmod 1023 > 0$:
 - FLIT_REPLAY_NUM is cleared.
 - ACKD_FLIT_SEQ_NUM is set to N.
 - If the received Flit was an Ack Flit :
 - NAK_IGNORE_FLIT_SEQ_NUM is cleared to 000h.
 - If $(N - \text{TX_REPLAY_FLIT_SEQ_NUM}) \bmod 1023 < \text{MAX_UNACKNOWLEDGED_FLITS}$:
 - Set TX_REPLAY_FLIT_SEQ_NUM to 000h
 - Set REPLAY_SCHEDULED to 0b
 - If the received Flit was a Nak Flit :
 - If $N = \text{NEXT_TX_FLIT_SEQ_NUM} - 1$:
 - NAK_IGNORE_FLIT_SEQ_NUM is set to N.
 - NAK_IGNORE_FLIT_SEQ_NUM is cleared to 000h if all of the following are true:
 - A valid Nak Flit is received with Flit Sequence Number N.
 - $N \neq \text{NAK_IGNORE_FLIT_SEQ_NUM}$
 - $N \neq \text{NEXT_TX_FLIT_SEQ_NUM} - 1$
 - It is recommended that NAK_IGNORE_FLIT_SEQ_NUM is cleared to 000h if all of the following are true:
 - A valid Standard Nak Flit is received with Flit Sequence Number N.

- N == NAK_IGNORE_FLIT_SEQ_NUM
- REPLAY_SCHEDULED_TYPE is SELECTIVE_REPLAY
- TX_REPLY_FLIT_SEQ_NUM == N + 1
- Schedule a replay as necessary, following the Flit Replay Scheduling rules in § Section 4.2.3.4.2.1.6.
 - The value of NAK_IGNORE_FLIT_SEQ_NUM must be updated before evaluating the rules in § Section 4.2.3.4.2.1.6.

4.2.3.4.2.1.5 Ack, Nak, and Discard Rules §

```

if ( NON_IDLE_EXPLICIT_SEQ_NUM_FLIT_RCV == 0b) {
    See Receiver Rules in IDLE Flit Handshake Phase
}

# Invalid Flit Handling
else if (Received Invalid Flit) {
    if (NAK_SCHEDULED == 1) {
        if (NAK_SCHEDULED_TYPE == STANDARD_NAK) {
            Flit Discard 1
        }
        else {
            Nak Schedule 6
        }
    }
    else {
        if (NAK_WITHDRAWAL_ALLOWED == 1) {
            Nak Schedule 1
        }
        else {
            Nak Schedule 0
        }
    }
}

# Valid Flit Handling
else if (Received Valid Flit) {
    if ((Explicit Sequence Number Flit) && (Flit Sequence Number == 0)) {
        Flit Discard 2
    }
    else if (NAK_WITHDRAWAL_ALLOWED == 1b) {
        NAK_WITHDRAWAL_ALLOWED = 0b;

        if (Prior Flit was Payload) {
            if (NOP Flit) {
                # Schedule a Standard or Selective Nak
                Discard Flit's TLP Bytes
                NAK_SCHEDULED = 1b;
                TX_ACKNAK_FLIT_SEQ_NUM = NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1;
                NAK_SCHEDULED_TYPE = STANDARD_NAK or SELECTIVE_NAK
                NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change;
            }
        }
    }
}

```

Base 6.4 vs Base 6.3

```

        if (Scheduling a Selective Nak) {
            NEXT_RX_FLIT_SEQ_NUM_TO_STORE = NEXT_EXPECTED_RX_FLIT_SEQ_NUM + 1
        }
    }
# Current Flit is Payload Flit
else {
    # Schedule a Standard Nak
    if (Scheduling a Standard Nak) {
        standard_nak_procedure()
    }
    # Schedule a Selective Nak
    else {
        selective_nak_procedure()
    }
}
# Prior Flit was NOP
else {
    if (bad_sequence_number() || ((NOP Flit) && bad_nop_sequence_number())) {
        Nak Schedule 2
    }
    # Withdraw Nak
    else {
        if (((Payload Flit) && duplicate_sequence_number()) || (NOP Flit)) {
            # Schedule Ack for NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1
            TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1
            Discard Flit's TLP Bytes
        }

        if ((Payload Flit) && (NEXT_EXPECTED_RX_FLIT_SEQ_NUM ==
IMPLICIT_RX_FLIT_SEQ_NUM)) {
            # Schedule Ack for NEXT_EXPECTED_RX_FLIT_SEQ_NUM
            TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM
            Increment NEXT_EXPECTED_RX_FLIT_SEQ_NUM
        }
    }
}
# NAK_WITHDRAWAL_ALLOWED == 0b
else {
    if (NAK_SCHEDULED == 1b) {
        if (NAK_SCHEDULED_TYPE == Standard Nak) {
            if (NOP Flit) {
                if ((Explicit Sequence Number Flit) &&
                    ( IMPLICIT_RX_FLIT_SEQ_NUM == ( NEXT_EXPECTED_RX_FLIT_SEQ_NUM -1)))
{
                    NAK_SCHEDULED = 0b
                    Discard Flit's TLP Bytes
                }
                else {
                    Flit Discard 0
                }
            }
        }
    }
}
# Payload Flit

```

Base 6.4 vs Base 6.3

```

        else {
            standard_nak_procedure()
        }
    }
# Selective Nak
else {
    if (NOP Flit) {
        Flit Discard 0
    }
# Payload Flit
else {
    selective_nak_procedure()
}
}
else {
    if (NOP Flit) {
        if (bad_nop_sequence_number())) {
            Nak Schedule 2
        }
        else {
            Flit Discard 0
        }
    }
# Payload Flit
else {
    if (duplicate_sequence_number()) {
        Flit Discard 0
    }
    else if (bad_sequence_number()){
        Nak Schedule 2
    }
    else {
        # Forward Progress
        Ack Schedule 0
    }
}
}
}
}

# Bad Sequence Number Check Logic
def bad_sequence_number() {
    return (NEXT_EXPECT_RX_FLIT_SEQ_NUM - IMPLICIT_RX_FLIT_SEQ_NUM) mod 1023 > 511
}

def bad_nop_sequence_number() {
    return bad_sequence_number() || (IMPLICIT_RX_FLIT_SEQ_NUM ==
NEXT_EXPECTED_RX_FLIT_SEQ_NUM)
}

# Duplicate Sequence Number Check Logic
def duplicate_sequence_number() {

```

Base 6.4 vs Base 6.3

```

        return ((TX_ACKNAK_FLIT_SEQ_NUM - IMPLICIT_RX_FLIT_SEQ_NUM) mod 1023 < 511)
    }

# Ack/Nak/Discard Logic when scheduling a Standard Nak or when a Standard Nak is
outstanding.
def standard_nak_procedure() {
    if (duplicate_sequence_number()) {
        if (NAK_SCHEDULED == 1b) {
            Flit Discard 0
        }
        else {
            Nak Schedule 2
        }
    }
    else if ((Received Flit is Explicit Sequence Number Flit) &&
              (IMPLICIT_RX_FLIT_SEQ_NUM == NEXT_EXPECTED_RX_FLIT_SEQ_NUM)) {
        Ack Schedule 1
    }
    else {
        if (NAK_SCHEDULED == 1b) {
            Flit Discard 0
        }
        else {
            Nak Schedule 2
        }
    }
}

# Ack/Nak/Discard Logic when scheduling a Selective Nak or when a Selective Nak is
outstanding.
def selective_nak_procedure() {
    if (duplicate_sequence_number()) {
        Flit Discard 0
    }
    else if ((Received Flit is Explicit Sequence Number Flit) &&
              (IMPLICIT_RX_FLIT_SEQ_NUM == NEXT_EXPECTED_RX_FLIT_SEQ_NUM)) {

        if (RX_RETRY_BUFFER_OVERFLOW == 1) {
            Nak Schedule 5
        }
        else {
            Ack Schedule 2
        }
    }
    else if (((IMPLICIT_RX_FLIT_SEQ_NUM == NEXT_RX_FLIT_SEQ_NUM_TO_STORE)
              && (NAK_SCHEDULED == 1))
              || ((IMPLICIT_RX_FLIT_SEQ_NUM == NEXT_EXPECTED_RX_FLIT_SEQ_NUM + 1)
                  && (NAK_SCHEDULED == 0))) {
        if (RX_RETRY_BUFFER_FULL) {
            Nak Schedule 4
        }
        else {
            Nak Schedule 3
        }
    }
}

```

```

    }
}
else {
    # RX Retry Buffer Overflow
    if ((Received Flit is Valid) && (RX_RETRY_BUFFER_OVERFLOW == 1)) {
        Either Flit Discard 0 or NAK Schedule 275
    }
    # Bad Sequence Number
    else {
        Nak Schedule 2
    }
}
}

```

Nak Schedule 0

- The received Flit is discarded.
- NAK_WITHDRAWAL_ALLOWED is set to 1b.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Nak Schedule 1

- The received Flit is discarded.
- NAK_SCHEDULED is set to 1b.
- TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1.
- NAK_SCHEDULED_TYPE is set to STANDARD_NAK.
- NAK_WITHDRAWAL_ALLOWED is cleared.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Nak Schedule 2

- The TLP Bytes of the received Flit are discarded.
- TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1.
- NAK_SCHEDULED is set to 1b.
- NAK_SCHEDULED_TYPE is set to STANDARD_NAK.
- RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM is set to 000h.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Nak Schedule 3

- If NAK_SCHEDULED is 0b
 - NAK_SCHEDULED is set to 1b.
 - NAK_SCHEDULED_TYPE is set to SELECTIVE_NAK
 - TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1
- Store the Received Flit in the RX Retry Buffer.
- RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM is set to IMPLICIT_RX_FLIT_SEQ_NUM.
- NEXT_RX_FLIT_SEQ_NUM_TO_STORE is set to IMPLICIT_RX_FLIT_SEQ_NUM + 1.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Nak Schedule 4

- The TLP Bytes of the received Flit that cannot be stored in the RX Retry Buffer are discarded.

⁷⁵. Flit Discard 0 is the preferred behavior in this scenario. Nak Schedule 2 is allowed but is less performant.

- If the Receiver chooses to continue with the Selective Nak :
 - RX_RETRY_BUFFER_OVERFLOW is set to 1b.
 - RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM does not change.
- If the Receiver chooses to change the Selective Nak to a Standard Nak :
 - NAK_SCHEDULED_TYPE is set to STANDARD_NAK.
 - TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1.
 - All Flits stored in the RX Retry Buffer are purged.
 - RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM is set to 000h.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Nak Schedule 5

- NAK_SCHEDULED remains 1b.
- NAK_SCHEDULED_TYPE is set to STANDARD_NAK.
- TX_ACKNAK_FLIT_SEQ_NUM is set to RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM.
- RX_RETRY_BUFFER_OVERFLOW is cleared.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM is set to RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM + 1.
- RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM is set to 000h.

Nak Schedule 6

- The received Flit is discarded.
- NAK_SCHEDULED remains 1b.
- NAK_SCHEDULED_TYPE is set to STANDARD_NAK.
- RX_RETRY_BUFFER_OVERFLOW is cleared.
- TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM - 1.
- All Flits stored in the RX Retry Buffer are purged.
- RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM is set to 000h.

Nak Schedule 7

- A Receiver is permitted to change a scheduled Selective Nak of TX_ACKNAK_FLIT_SEQ_NUM to a Standard Nak of TX_ACKNAK_FLIT_SEQ_NUM for any reason.

Ack Schedule 0

- TX_ACKNAK_FLIT_SEQ_NUM is set to NEXT_EXPECTED_RX_FLIT_SEQ_NUM.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM is incremented.

Ack Schedule 1

- NAK_SCHEDULED is cleared.
- TX_ACKNAK_FLIT_SEQ_NUM is set to IMPLICIT_RX_FLIT_SEQ_NUM.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM is set to IMPLICIT_RX_FLIT_SEQ_NUM + 1.

Ack Schedule 2

- NAK_SCHEDULED is cleared.
- If RX Retry Buffer is empty:
 - TX_ACKNAK_FLIT_SEQ_NUM is set to IMPLICIT_RX_FLIT_SEQ_NUM.
 - NEXT_EXPECTED_RX_FLIT_SEQ_NUM is set to IMPLICIT_RX_FLIT_SEQ_NUM + 1.

- If RX Retry Buffer is NOT empty:
 - TX_ACKNAK_FLIT_SEQ_NUM is set to RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM .
 - NEXT_EXPECTED_RX_FLIT_SEQ_NUM is set to RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM + 1.
- RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM is set to 000h.

Flit Discard 0

- The TLP Bytes of the received Flit are discarded.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Flit Discard 1

- The received Flit is discarded.
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.

Flit Discard 2

- The received Flit is discarded.
- A Data Link Protocol Error is logged in the receiving Port (see § Section 6.2).
- NEXT_EXPECTED_RX_FLIT_SEQ_NUM does not change.
- NAK_WITHDRAWAL_ALLOWED is cleared.

Base 6.4 vs Base 6.3

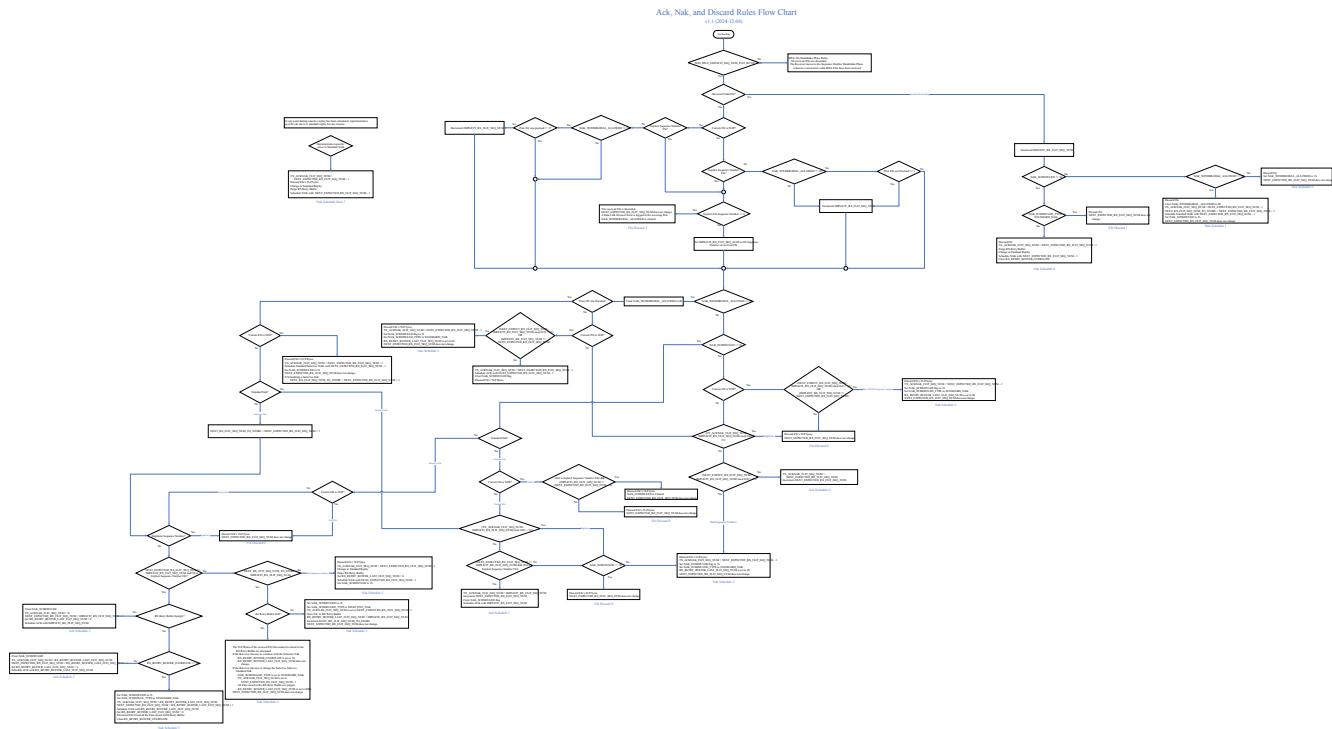


Figure 4-33 Flit Ack, Nak, and Discard Rules Flow Chart (Zoom-In to View) §

4.2.3.4.2.1.6 Flit Replay Scheduling §

Replay Schedule Rule 0

Replay Timeout

- If all of the following are true:
 - REPLAY_TIMEOUT_FLIT_COUNT ≥ 1500 ⁷⁶
 - REPLAY_SCHEDULED is 0b.
 - REPLAY_IN_PROGRESS is 0b.
- Then:
 - REPLAY_SCHEDULED is set to 1b.
 - REPLAY_SCHEDULED_TYPE is set to STANDARD_REPLAY.
 - TX_REPLAY_FLIT_SEQ_NUM is set to (ACKD_FLIT_SEQ_NUM + 1).
 - CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is set to 0b.⁷⁷

76. The max Ordered set insertion interval in § Table 4-32 is 748. 1500 is 748×2 rounded up.

- CONSECUTIVE_TX_NAK_FLITS is set to 0b.⁷⁸
- REPLAY_TIMEOUT_FLIT_COUNT is set to 0b.
- A Replay Timer Timeout error is logged in the Port (see § Section 6.2).

Replay Schedule Rule 1

Received Nak for Payload Flit.

- If all of the following are true:
 - Received a valid Nak Flit with Flit Sequence Number N.
 - A Payload Flit with Flit Sequence Number N + 1 is stored in the TX Retry Buffer.
 - REPLAY_SCHEDULED is 0b.
 - REPLAY_IN_PROGRESS is 0b.
 - Either of the following are true:
 - NAK_IGNORE_FLIT_SEQ_NUM != N.
 - The Nak Ignore Window has not been started or has elapsed.
- Then:
 - REPLAY_SCHEDULED is set to 1b.
 - TX_REPLAY_FLIT_SEQ_NUM is set to N + 1.
 - NAK_IGNORE_FLIT_SEQ_NUM is set to N.
 - If the received Flit is Standard Nak Flit :
 - REPLAY_SCHEDULED_TYPE is set to STANDARD_REPLY.
 - If the received Flit is a Selective Nak Flit :
 - REPLAY_SCHEDULED_TYPE is set to SELECTIVE_REPLY.

Replay Schedule Rule 2

Received Standard Nak after Selective Nak for Payload Flit.

- If all of the following are true:
 - Received a valid Standard Nak Flit with Flit Sequence Number N.
 - N+1 equals TX_REPLAY_FLIT_SEQ_NUM.
 - REPLAY_SCHEDULED is 1b.
 - REPLAY_IN_PROGRESS is 0b.
 - REPLAY_SCHEDULED_TYPE is SELECTIVE_REPLY.
- Then:
 - REPLAY_SCHEDULED is set to 1b.
 - TX_REPLAY_FLIT_SEQ_NUM does not change.
 - REPLAY_SCHEDULED_TYPE is set to STANDARD_REPLY.

4.2.3.4.2.1.7 Flit Replay Transmit Rules §

Flit Replay Transmit Rule 0

Standard Replay Scenario.

- If all of the following conditions are true:

77. CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS is cleared to allow the Transmitter to begin replaying Flits right away and avoid overflowing the TX Retry Buffer sending NOP Flits while waiting to send another Explicit Sequence Number Flit to begin the replay.

78. Same reason as clearing CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS above.

- REPLAY_SCHEDULED is 1b.
- REPLAY_IN_PROGRESS is 0b.
- REPLAY_SCHEDULED_TYPE is STANDARD_REPLAY.
- Either of the following are true:
 - The transmitter is in the Sequence Number Handshake Phase.
 - CONSECUTIVE_TX_NAK_FLITS is either 0 or greater than 2.
- Either of the following are true:
 - Data rate is $\frac{1}{2} \downarrow 64.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 64.0 \text{ GT/s} \uparrow$ and FLIT_REPLAY_NUM < 111b.
 - Data rate $\leq \frac{1}{2} \downarrow 32.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 32.0 \text{ GT/s} \uparrow$ and FLIT_REPLAY_NUM < 110b.
- Then:
 - The Transmitter must replay all unacknowledged Flits from the TX Retry Buffer beginning with the Flit containing Flit Sequence Number TX_REPLAY_FLIT_SEQ_NUM.
 - If the data rate is $\frac{1}{2} \downarrow 64.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 64.0 \text{ GT/s} \uparrow$
 - FLIT_REPLAY_NUM is incremented by 1.
 - If the data rate $\leq \frac{1}{2} \downarrow 32.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 32.0 \text{ GT/s} \uparrow$
 - FLIT_REPLAY_NUM is incremented by 2^{79} .
 - REPLAY_IN_PROGRESS is set to 1b.
 - REPLAY_SCHEDULED is set to 0b.

Flit Replay Transmit Rule 1

Selective Replay Scenario.

- If all of the following conditions are true:
 - REPLAY_SCHEDULED is 1b.
 - REPLAY_IN_PROGRESS is 0b.
 - REPLAY_SCHEDULED_TYPE is SELECTIVE_REPLAY.
 - Either of the following are true:
 - The transmitter is in the Sequence Number Handshake Phase.
 - CONSECUTIVE_TX_NAK_FLITS is either 0 or greater than 2.
 - Either of the following are true:
 - Data rate is $\frac{1}{2} \downarrow 64.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 64.0 \text{ GT/s} \uparrow$ and FLIT_REPLAY_NUM < 111b.
 - Data rate $\leq \frac{1}{2} \downarrow 32.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 32.0 \text{ GT/s} \uparrow$ and FLIT_REPLAY_NUM < 110b.
- Then:
 - The Transmitter must replay the Flit from the TX Retry Buffer which contains the Flit Sequence Number that matches TX_REPLAY_FLIT_SEQ_NUM.
 - If the data rate is $\frac{1}{2} \downarrow 64.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 64.0 \text{ GT/s} \uparrow$
 - FLIT_REPLAY_NUM is incremented by 1.
 - If the data rate $\leq \frac{1}{2} \downarrow 32.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 32.0 \text{ GT/s} \uparrow$
 - FLIT_REPLAY_NUM is incremented by 2^{80} .
 - REPLAY_IN_PROGRESS is set to 1b.
 - REPLAY_SCHEDULED is set to 0b.

79. Replays should happen less often at speeds lower than $\frac{1}{2} \downarrow 64.0 \text{ GT/s} \downarrow \frac{1}{2} \uparrow 64.0 \text{ GT/s} \uparrow$. We increment FLIT_REPLAY_NUM by 2 so that the Replay Rollover will happen after fewer Replays.

80. Previous versions of this specification had this as increment by 1. This remains permitted, although not optimal, behavior.

Flit Replay Transmit Rule 2

Replay Rollover Scenario.

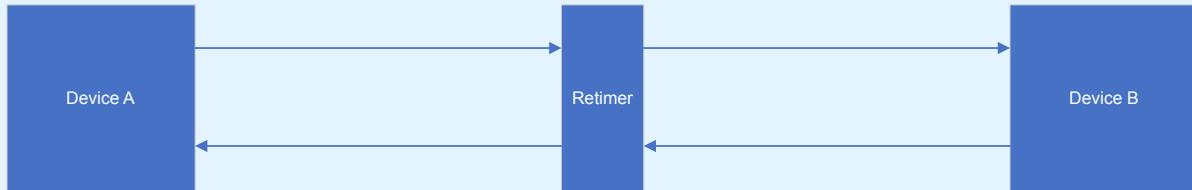
- If all of the following conditions are true:
 - REPLAY_SCHEDULED is 1b.
 - REPLAY_IN_PROGRESS is 0b.
 - Either of the following are true:
 - Data rate is $\downarrow\downarrow 64.0 \text{ GT/s} \downarrow\downarrow$ $\uparrow\uparrow 64.0 \text{ GT/s} \uparrow\uparrow$ and FLIT_REPLAY_NUM equals 111b.
 - Data rate $\leq \downarrow\downarrow 32.0 \text{ GT/s} \downarrow\downarrow$ $\uparrow\uparrow 32.0 \text{ GT/s} \uparrow\uparrow$ and FLIT_REPLAY_NUM $\geq 110b$.
- Then:
 - If the data rate is $\downarrow\downarrow 64.0 \text{ GT/s} \downarrow\downarrow$ $\uparrow\uparrow 64.0 \text{ GT/s} \uparrow\uparrow$
 - FLIT_REPLAY_NUM is incremented by 1 (causing a rollover to 000b).
 - If the data rate $\leq \downarrow\downarrow 32.0 \text{ GT/s} \downarrow\downarrow$ $\uparrow\uparrow 32.0 \text{ GT/s} \uparrow\uparrow$
 - FLIT_REPLAY_NUM is incremented by 2 (causing a rollover to 000b or 001b).
 - A REPLAY_NUM Rollover error is logged in the Port (see § Section 6.2).
 - The Port must enter Recovery.

IMPLEMENTATION NOTE: FLIT ACK/NAK/REPLAY EXAMPLE §

Consider two devices A and B connected through a Retimer, as shown in § Figure 4-34.

1. Both Device A and Device B are in the Normal Flit Exchange Phase
2. B receives valid Payload Flits 1 through 10 from A. B sends Acks for those Flits back to A. (Ack Schedule 0)
3. B receives an invalid NOP Flit 10⁸¹ and sets NAK_WITHDRAWAL_ALLOWED to 1b. (Nak Schedule 0)
4. B then receives a valid Flit 11 with Prior Flit was Payload set to 0b, indicating that prior Flit was a NOP Flit. B sends an Ack with Flit Sequence Number 11 to A.
 - NAK_WITHDRAWAL_ALLOWED is cleared and a Nak is never sent for the invalid NOP Flit 10.
5. B then receives an invalid Payload Flit 12 and sends sets NAK_WITHDRAWAL_ALLOWED to 1b. (Nak Schedule 0)
6. B then receives a valid Flit 13 that indicates that Flit 12 was a Payload Flit, and sends a Selective Nak Flit with Flit Sequence Number 11, requesting A to Replay only Flit 12. (Nak Schedule 1)
 - B will then send a minimum of three consecutive Selective Nak Flits with Flit Sequence Number 11.
 - If B was sending Explicit Sequence Number Flits when the Nak of invalid Flit 12 was scheduled, it would have held off sending the Nak Flit until it had finished sending 3 consecutive Explicit Sequence Number Flits (Replay Command Rules in Normal Flit Exchange Phase).
 - The choice of a Selective Nak by B is an optimization in this case. B has enough space in its RX Retry Buffer to store the subsequent Flits.
 - If B were to receive an invalid Flit with a Selective Nak outstanding, it would be required to change the outstanding Selective Nak to a Standard Nak. (Nak Schedule 6)
 - B is also permitted to change the outstanding Selective Nak to a Standard Nak for any implementation specific reason. (Nak Schedule 7)
7. After B sends 3 consecutive Nak Flits with Flit Sequence Number 11, B will continue following the Replay Command Rules in Normal Flit Exchange Phase to determine whether to send Nak Flits or Explicit Sequence Number Flits.
8. In the time it took for A to receive the Selective Nak of Flit 12 and schedule a Replay of Flit 12, valid Payload Flits 13 and 14 and NOP Flit 14 were received by B. B stores Payload Flits 13 and 14 in its RX Retry Buffer. The TLP Bytes of NOP Flit 14 are discarded and the Flit is not stored in the RX Retry Buffer (Flit Discard 0).
9. B then receives replayed Flit 12 from A and processes Flit 12 as well as Payload Flits 13 and 14 that were stored in the B's RX Retry Buffer.
 - Note that Payload Flits 12 and 15 and NOP Flit 15 are all sent with explicit Flit Sequence Numbers in adherence to the Replay Command Rules in Normal Flit Exchange Phase
10. Next, B receives valid Payload Flit 15, and then invalid NOP Flit 15. B sets NAK_WITHDRAWAL_ALLOWED to 1b.
11. B then receives invalid Flit 16, and since there's no way for B to know that the previous invalid Flit was a NOP Flit, the Nak can't be withdrawn and B sends a Standard Nak with Flit Sequence Number 15.
 - Note that B cannot send a Selective Nak since two consecutive invalid Flits were received.

12. B then receives valid Payload Flits 17 and 18, but since B has a Standard Nak outstanding, the TLP Bytes of these Flits are discarded (Flit Discard 0).
13. A then replays all unacknowledged Flits from its TX Retry Buffer (Flits 16, 17, and 18), which B receives as valid Flits and Ack accordingly.



$t = N + 16 \text{ Flit Times}$ —————→ $t = N$

| | | | | | | | | | | | | | | | |
|----------------------------|----------------------------|----------------------------|---------------------------|---------------------------|---------------------------|----------------------------|----------------------------|----------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|
| Payload Flit 18 ✓ | Payload Flit 17 ✓ | Payload Flit 16 ✓ | Payload Flit 18 ✓ | Payload Flit 17 ✓ | Payload Flit 16 ✗ | NOP Flit 15 ✗ | Payload Flit 15 ✓ | NOP Flit 14 ✓ | Payload Flit 12 ✓ | Payload Flit 14 ✓ | Payload Flit 13 ✓ | Payload Flit 12 ✗ | Payload Flit 11 ✓ | NOP Flit 10 ✗ | Payload Flit 10 ✓ |
| Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 00b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 00b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 01b | Flit Usage: 00b | Flit Usage: 01b |
| Prior Flit: Payload | Prior Flit: Payload | Prior Flit: Payload | Prior Flit: Payload | Prior Flit: Payload | Prior Flit: NOP | Prior Flit: NOP | Prior Flit: Payload | Prior Flit: NOP | Prior Flit: Payload | Prior Flit: NOP | Prior Flit: Payload | Prior Flit: NOP | Prior Flit: Payload | Prior Flit: NOP | Prior Flit: Payload |
| Replay Cmd: Explicit | Replay Cmd: Explicit | Replay Cmd: Explicit | Replay Cmd: Ack/Nak | Replay Cmd: Ack/Nak | Replay Cmd: Ack/Nak | Replay Cmd: Explicit | Replay Cmd: Explicit | Replay Cmd: Explicit | Replay Cmd: Ack/Nak |

Figure 4-34 Flit Ack/Nak/Replay Example §

81. NOP Flits do not consume a Flit Sequence Number and just use the Flit Sequence Number as described in Flit Sequence Number and Retry Mechanism (see § Section 4.2.3.4.2.1).

IMPLEMENTATION NOTE: TRANSMITTER THROTTLING AFTER SELECTIVE REPLAY §

In some situations, it will reduce link latency if the transmitter throttles its Flit transmission rate after sending a Selective Replay Flit.

When the Receiver issues a Selective Nak and is waiting for the selective replay, it stores valid Payload Flits in an RX retry buffer and processes these Flits after the selective replay is received. Payload Flits that arrive after the selective replay must also be stored in the RX retry buffer. All Payload Flits are processed in sequence number order.

Without transmitter throttling, this RX retry buffer adds latency to the receive path since all the entries must be consumed in order. Latency can be reduced and this performance impact can be mitigated if the transmitter throttles Flit transmission earlier allowing the RX retry buffer to drain. This early transmitter throttling does not affect overall performance – the TLPs that get delayed at the transmitter would have been delayed at the Receiver. Performing early throttling at the transmitter allows the RX retry buffer to drain so that subsequent TLPs have reduced latency.

Throttling is only needed when the Link is operating at the drain rate of the Receiver. If the Link is operating at a reduced width (e.g., due to L0p), throttling may not be needed as the RX retry buffer drain rate may be faster than the Link rate.

To compute how to throttle, the Transmitter can assume that the initial link width is the drain rate of the Receiver. On receiving a selective Nak, the Transmitter can compute the number of Payload Flits in the Receiver's RX retry buffer (or on the way to that buffer) since it knows how many Payload Flits are currently in its TX retry buffer. The transmitter should throttle and/or enforce a certain bandwidth efficiency (i.e., restrict the number of NOP TLPs in a Payload Flit) in each new Payload Flit it transmits from its Transaction Layer to allow the Receiver's RX retry buffer to drain.

For example, suppose a x8 Link receives a selective Nak for Flit number X and the Transmitter has sent 4 Payload Flits after it initially sent Flit number X. In this case, the transmitter should throttle its upper layers to create a 4 Flit “bubble”. For a x8 Link, Flits are 16 Symbols and SKP Ordered Sets are 32 Symbols so this “bubble” is either 4 NOP Flits, 2 NOP Flits and 1 SKP Ordered Set, or 2 SKP Ordered Sets. One mechanism to implement the throttling is for the transmitter to enforce <Y% NOP TLPs (e.g., Y = 25) in its Payload Flits until it has sent the “bubble” (if TLPs are constantly being transmitted this could take 2 SKP Ordered Sets).

↑↑Three NOP Flit types have been defined in § Table 4-22, along with their intended usages. All types defined below are considered NOP Flits and must follow all NOP Flit rules outlined in this specification. Refer to § Section 6.37.1 for the usage model of NOP.Debug and NOP.Vendor Flit types.↑

↑↑The NOP Flit types defined below may rely on information provided by upper layers, but the transmission is done solely by the Physical Layer independently of the upper layers. Transmission may begin as soon as the Link enters L0. For NOP.Empty and NOP.Debug Flits, no credit checks are required for transmission. For NOP.Vendor Flits, implementation of any credit mechanism and associated checking for the credits is implementation specific. A receiver that comprehends non-zero encodings must silently drop any NOP.Debug or NOP.Vendor Flits if it does not support processing them. Receiver behavior for designs that do support processing NOP.Debug or NOP.Vendor Flits is implementation specific but transmitting/receiving NOP.Debug and NOP.Vendor Flits must not affect the link state.↑

Table 4-22 ↑↑NOP Flit Types §

| ↑↑NOP Flit Type↑ | ↑↑Usage↑ |
|------------------|---|
| ↑↑NOP.Empty↑ | ↑↑Empty payload (see § Section 4.2.3.4.3.1)↑ |

| | |
|-------------------------|--|
| ↑↑NOP Flit Type↑ | ↑↑Usage↑ |
| ↑↑NOP.Debug↑ | ↑↑Deliver debug information (see § Section 4.2.3.4.3.2)↑ |
| ↑↑NOP.Vendor↑ | ↑↑Deliver vendor-defined information (see § Section 4.2.3.4.3.3)↑ |

↑↑All NOP Flit TLPs use a common definition of the first DW of the NOP Flit Payload. Subsequent content depends on the value of NOP Flit Type (see § Figure 4-35).↑

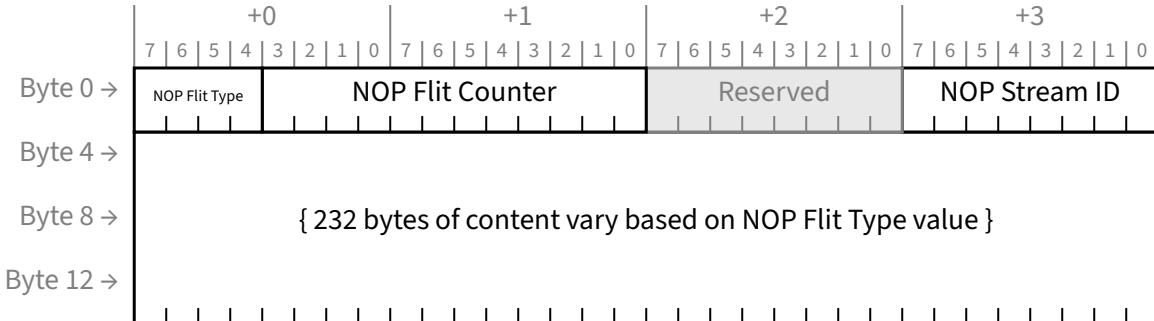


Figure 4-35 ↑↑NOP Flit Common Header↑ §

Table 4-23 ↑↑NOP Flit Common Header Fields↑ §

| ↑↑Field↑ | ↑↑Location↑ | ↑↑Definition↑ | |
|----------------------------|---|--|---|
| ↑↑NOP Flit Type↑ | ↑↑Byte 0: Bits 7:4↑ | ↑↑0h↑ | ↑↑NOP.Empty Flit↑ |
| | | ↑↑1h↑ | ↑↑NOP.Debug Flit↑ |
| | | ↑↑2h to Eh↑ | ↑↑Reserved. Receiver must treat as a NOP.Empty Flit↑ |
| | | ↑↑Fh↑ | ↑↑NOP.Vendor Flit↑ |
| ↑↑NOP Flit Counter↑ | ↑↑{ Byte 0: Bits 3:0,↑ ↑↑Byte 1: Bits 7:0 }↑ | ↑↑NOP.Empty Flit↑ | ↑↑Transmitter populates with 000h, Receiver is permitted to ignore↑ |
| | | ↑↑NOP.Debug Flit and NOP.Vendor Flit↑ | ↑↑Incrementing per-NOP Flit Type counter of NOP.Debug and NOP.Vendor Flit types↑ |
| ↑↑Reserved↑ | ↑↑Byte 2: Bits 7:0↑ | ↑↑Reserved↑ | |
| ↑↑NOP Stream ID↑ | ↑↑Byte 3: Bits 7:0↑ | ↑↑NOP.Empty Flit↑ | ↑↑Transmitter populates with 00h, Receiver is permitted to ignore↑ |
| | | ↑↑NOP.Debug Flit and NOP.Vendor Flit↑ | ↑↑Original source identifier of the NOP Flit↑ |

↑↑A non-empty NOP Flit is a NOP Flit with a non-zero NOP Flit Type encoding.↑

↑↑A NOP Stream is a sequence of NOP Flits that are sourced by a Transmitter. The NOP Stream ID field in the NOP Flit provides a mechanism to identify the original Transmitter of the NOP Flit.↑

- ↑↑The NOP Flit Extended Capability structure (see § Section 7.8.14) provides software control (via the NOP Stream ID Start and Number of NOP Streams fields) over the range of values that a Transmitter may use. The usage details of the programmed range are implementation specific.↑

- ↑↑↑The combination of NOP Flit Type value and NOP Stream ID value uniquely identifies a NOP Stream and its value must be unique across all entities transmitting NOP Flits over a given Link.↑
- ↑↑↑NOP.Empty Flits must always transmit a NOP Stream ID value of 00h.↑

↑↑↑The NOP Flit Counter field in the NOP Flit allows dropped or missing non-empty NOP Flits to be detected.↑

- ↑↑↑Every NOP Stream must have an independent counter.↑
- ↑↑↑Each NOP Stream must start with a Nop Flit Counter value of 000h and every subsequent non-empty NOP Flit of the NOP Stream that is transmitted must increment the counter by one. The NOP Flit Counter must roll-over after FFEh.↑
 - ↑↑↑A NOP Flit Counter value of FFFh indicates an error or unexpected behavior occurred. This provides an indication that an unknown number of Flits of a NOP Stream were lost. This indication is only a hint since the Flit containing this indication could itself be lost.↑
- ↑↑↑When a NOP Stream is disabled, its NOP Flit Counter must reset to 000h.↑
- ↑↑↑NOP.Empty Flits must always transmit a NOP Flit Counter value of 000h.↑

↑↑↑Note: Receivers may forward received NOP Flits with a NOP Stream ID other than FFh to a different Link. In this scenario, the forwarded NOP Flit must preserve all fields as received, including the NOP Flit Type , the NOP Flit Counter and the NOP Stream ID . The forwarding Receiver is permitted to overwrite the NOP Flit Counter to FFFh in certain situations. The forwarding mechanism is implementation specific on a best-effort basis, with no guarantee of delivery. § Section 6.37.1 .↑

↑↑↑After the first DW, the remaining TLP Bytes carry the NOP Flit Payload, whose definition varies depending on the NOP Flit Type . See § Section 4.2.3.4.3 for NOP Flit Payload definitions.↑

4.2.3.4.3 NOP Flit Payload §

4.2.3.4.3.1 NOP.Empty Flit §

A NOP.Empty Flit has an empty NOP Flit Payload with all bytes set to 00h (see § Figure 4-36).

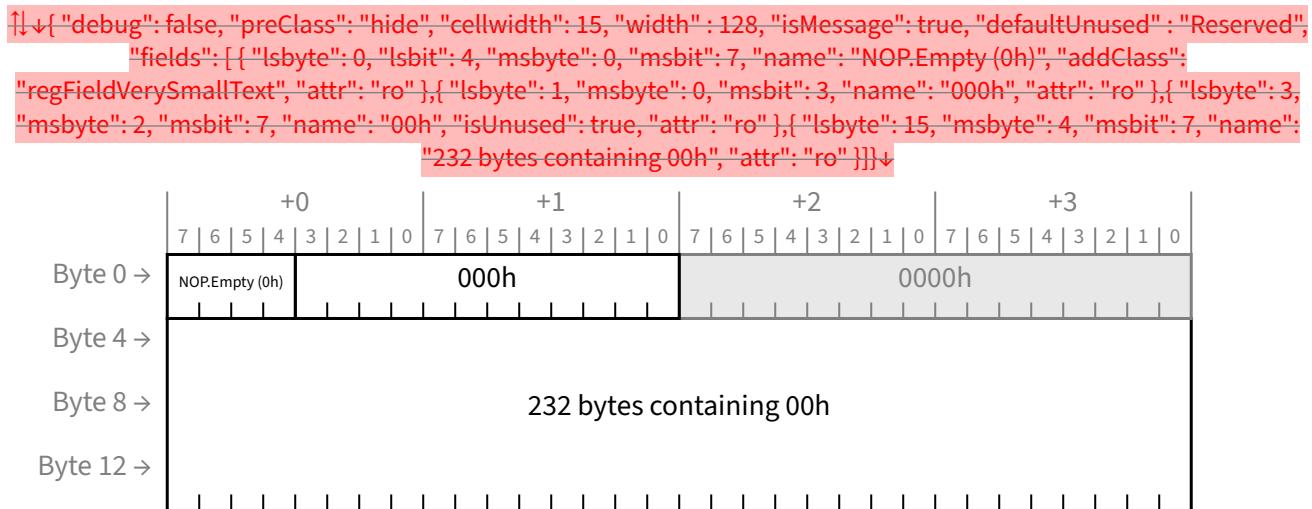


Figure 4-36 NOP.Empty Flit Payload §

4.2.3.4.3.2 NOP.Debug Flit §

NOP.Debug Flits contain a list of zero or more Debug Chunks. The first Debug Chunk starts at Byte 4 of the Flit (immediately after the NOP Flit Common Header, see § Figure 4-35).

A Debug Chunk consists of a variable-sized Debug Header followed by a vendor-defined variable-sized Debug Payload (see § Figure 4-37).

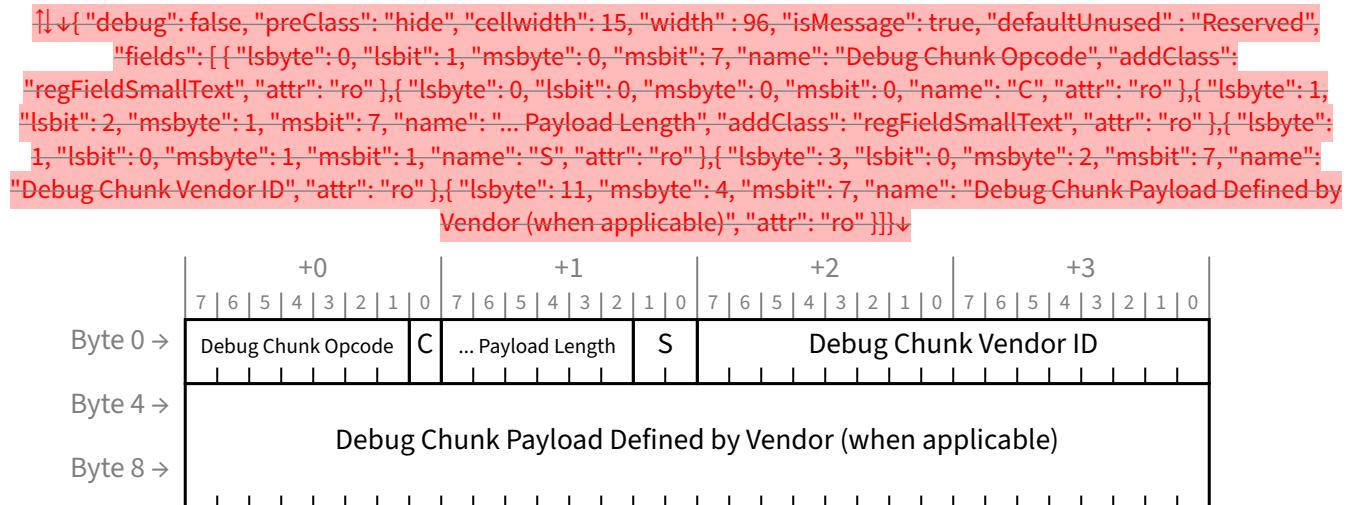


Figure 4-37 NOP.Debug Flit Debug Chunk §

Table 4-24 NOP.Debug Flit Debug Chunk Fields §

| Field | Location | Definition |
|-------------------------------------|--|---|
| Debug Chunk Opcode | Byte 0: Bits 7:1 | Debug opcode encoding defined by vendor that describes the debug content. |
| Debug Chunk Continuation (C) | Byte 0: Bit 0 | Indication this Debug Chunk is a continuation from the previous Debug Chunk. This is only valid for the first Debug Chunk of a Flit and must be Reserved otherwise. Continuation chunks must have the same <u>Debug Chunk Opcode</u> and <u>Debug Chunk Vendor ID</u> as the previous chunk. |
| Debug Chunk Payload Length | Byte 1: Bits 7:2 | Length of the Debug Chunk Payload of the current Debug Chunk in DW. The length cannot exceed the remaining number of DW in the 236 Byte TLP Bytes of the Flit. The maximum Length value is 57 (1 DW NOP Flit Common Header + 1 DW Debug Header + 57 DW Debug Chunk Payload = 236 Bytes of NOP Flit Payload). A Length value of 0 indicates this Debug Chunk only contains a Debug Header. |
| Debug Chunk Header Size (S) | Byte 1: Bits 1:0 | Total size of the Debug Chunk Header in DW. Any additional bits beyond the first DW of the Debug Chunk Header are Reserved. 00b One DW 01b Two DW 10b Four DW 11b Reserved. Receiver must treat any NOP.Debug Flits with this encoding as a NOP.Empty Flit. Receivers that support NOP.Debug Flits must handle all valid Debug Chunk Header Sizes. |
| Debug Chunk Vendor ID | { Byte 2: Bits 7:0, Byte 3: Bits 7:0 } | Vendor ID associated with the vendor that defined the Debug Opcode For PCI-SIG defined Debug Opcodes, this field must use the PCI-SIG Vendor ID (0001h) |

```

    "fields": [ { "lsbyte": 0, "lsbit": 1, "msbyte": 0, "msbit": 7, "name": "Debug Chunk Opcode", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "C", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 7, "name": "Payload Length = 1h", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "S=00b", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "Debug Chunk Vendor ID", "attr": "ro" }, { "lsbyte": 7, "msbyte": 4, "msbit": 7, "name": "Debug Chunk Payload Defined by Vendor", "attr": "ro" } ] } ]
  
```

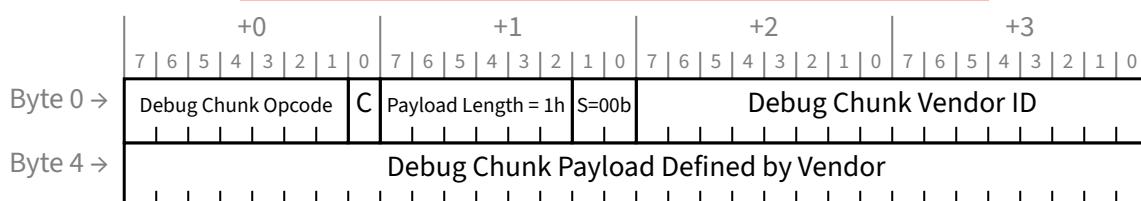


Figure 4-38 Example Debug Chunk with one DW Debug Chunk Header and one DW of Debug Chunk Payload §

Base 6.4 vs Base 6.3

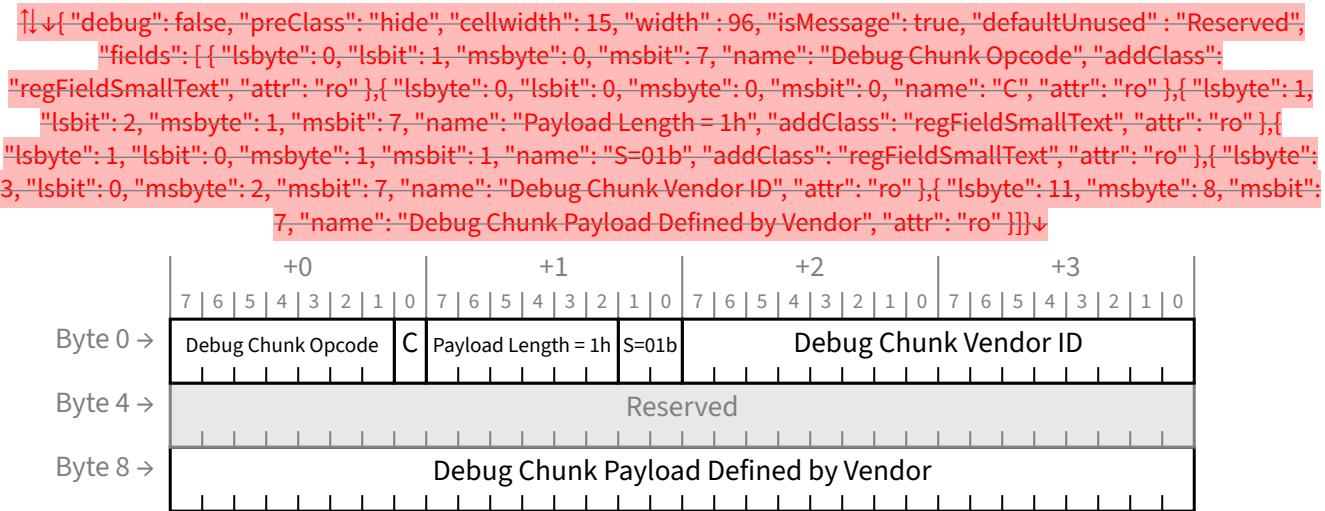


Figure 4-39 Example Debug Chunk with two DW Debug Chunk Header and one DW of Debug Chunk Payload §

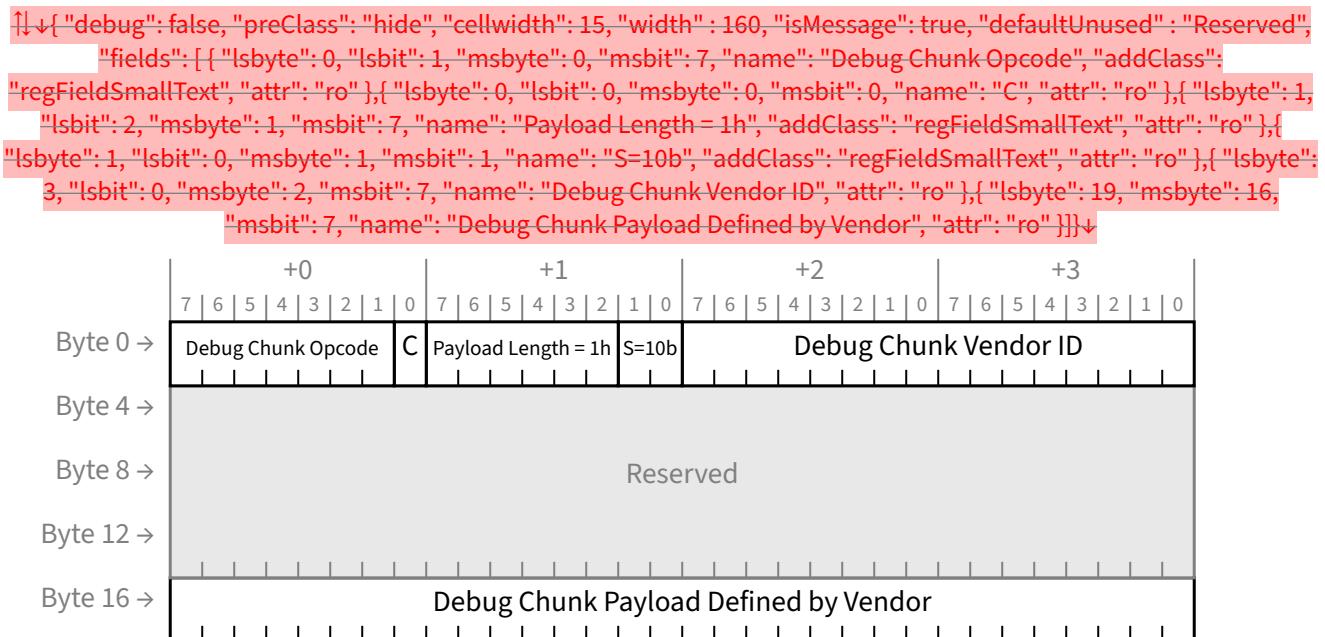


Figure 4-40 Example Debug Chunk with four DW Debug Chunk Header and one DW of Debug Chunk Payload §

A NOP.Debug Flit uses one or more Debug Chunks to deliver vendor-defined link debug information. § Figure 4-41 shows a NOP.Debug Flit with a single Debug Chunk.

Base 6.4 vs Base 6.3

```

↑↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "Reserved",
      "fields": [ { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "NOP.Debug (1h)", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "msbyte": 0, "msbit": 3, "name": "NOP Flit Counter", "attr": "ro" }, { "lsbyte": 3, "msbyte": 3, "msbit": 7, "name": "NOP Stream ID", "isUnused": true, "attr": "ro" }, { "lsbyte": 4, "lsbit": 1, "msbyte": 4, "msbit": 7, "name": "Debug Chunk Opcode", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 0, "name": "C", "attr": "ro" }, { "lsbyte": 5, "lsbit": 2, "msbyte": 5, "msbit": 7, "name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 5, "msbit": 1, "name": "S=00b", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Debug Chunk Vendor ID", "attr": "ro" }, { "lsbyte": 15, "msbyte": 8, "msbit": 7, "name": "Debug Chunk Payload Defined by Vendor", "attr": "ro" } ] }↓

```

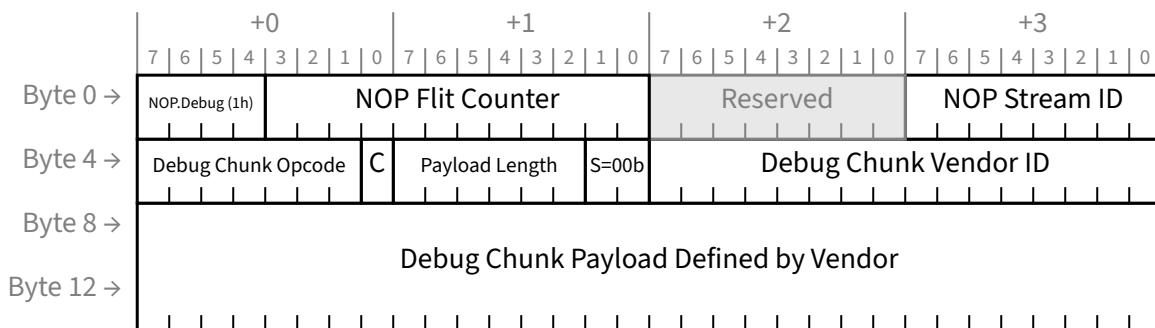


Figure 4-41 Example NOP.Debug Flit Payload with a single Debug Chunk with a one DW Debug Chunk Header §

If multiple Debug Chunks are inserted into a Flit, subsequent Debug Chunk Headers must begin immediately on the first available DW slot and set the Debug Chunk Continuation bit to 0b. It is permissible for Debug Chunks with different Debug Chunk Vendor ID values to be inserted into a single Flit. § Figure 4-42 shows a NOP.Debug Flit with multiple Debug Chunks.

```
↓↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 192, "isMessage": true, "defaultUnused": "Reserved", "fields": [ {"lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "NOP.Debug (1h)", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "msbyte": 0, "msbit": 3, "name": "NOP.Flit Counter", "attr": "ro"}, {"lsbyte": 3, "msbyte": 3, "msbit": 7, "name": "NOP.Stream ID", "isUnused": true, "attr": "ro"}, {"lsbyte": 4, "lsbit": 1, "msbyte": 4, "msbit": 7, "name": "Debug Chunk Opcode", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 4, "lsbit": 0, "msbyte": 4, "msbit": 0, "name": "C", "attr": "ro"}, {"lsbyte": 5, "lsbit": 2, "msbyte": 5, "msbit": 7, "name": "Payload Length = 1h", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 5, "msbit": 1, "name": "S=00b", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 6, "msbit": 7, "name": "1 DW of Debug Chunk Payload Defined by Vendor", "attr": "ro"}, {"lsbyte": 12, "lsbit": 1, "msbyte": 12, "msbit": 7, "name": "Debug Chunk Opcode", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 12, "lsbit": 0, "msbyte": 12, "msbit": 0, "name": "R", "isUnused": true, "attr": "ro"}, {"lsbyte": 13, "lsbit": 2, "msbyte": 13, "msbit": 7, "name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 13, "lsbit": 0, "msbyte": 13, "msbit": 1, "name": "S=00b", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 14, "msbit": 7, "name": "Debug Chunk Vendor ID", "attr": "ro"}, {"lsbyte": 23, "msbyte": 16, "msbit": 7, "name": "Debug Chunk Payload Defined by Vendor", "attr": "ro"} ]}
```

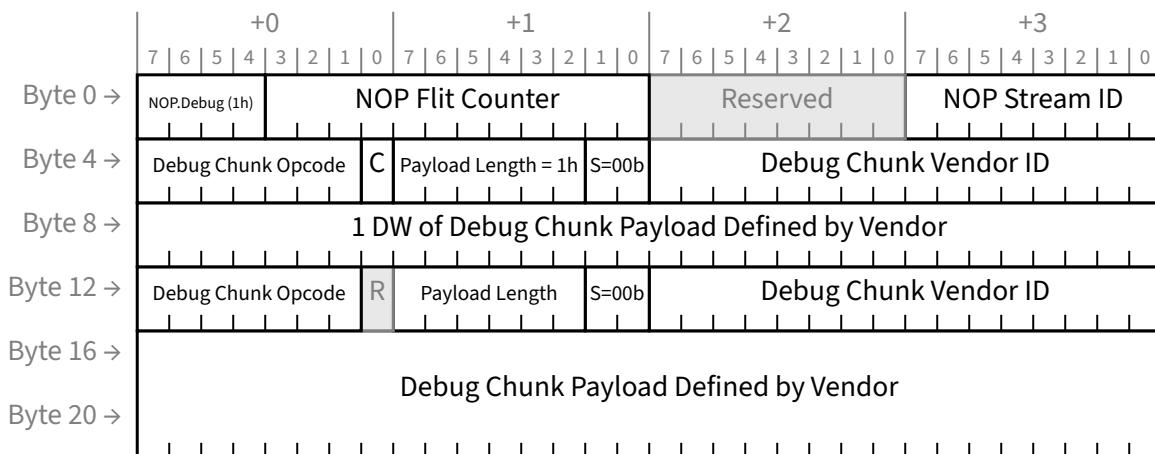


Figure 4-42 Example NOP;Debug Flit Payload with multiple Debug Chunks with one DW Debug Chunk Headers

Any unused DW slots at the end of the NOP Flit Payload of a NOP.Debug Flit must be filled with Empty Debug Chunks or set to 0.

4.2.3.4.3.2.1 PCI-SIG Defined Debug Chunk Opcode Values §

Table 4-25 PCI-SIG Defined Debug Chunk Opcode Values §

| Debug Chunk Opcode Value | Name | Description |
|-----------------------------|--|---|
| 000 0000b | <u>Empty Debug Chunk</u> | Padding and alignment, contains no valid debug information. See § Section 4.2.3.4.3.2.2 |
| 000 0001b | <u>Start Capture Trigger Debug Chunk</u> | Generic indication to start trace capture |
| 000 0010b | <u>Stop Capture Trigger Debug Chunk</u> | Generic indication to stop trace capture |
| 000 0011b | <u>FC Information Tracked by Transmitter Debug Chunk</u> | Flow Control information tracked by TX for TLP Transmission gating |

| Debug Chunk Opcode Value | Name | Description |
|-----------------------------|---|--|
| 000 0100b | <u>FC Information Tracked by Receiver Debug Chunk</u> | Flow Control information tracked by RX for TLP Receiver accounting |
| 000 0101b | <u>Flit Mode Transmitter Retry Flags and Counters Debug Chunk</u> | Transmitter Flag and Counter values used for Flit Sequence Number and retry mechanism in Flit Mode |
| 000 0110b | <u>Flit Mode Receiver Retry Flags and Counters Debug Chunk</u> | Receiver Flag and Counter values used for Flit Sequence Number and retry mechanism in Flit Mode |
| 000 0111b | <u>Buffer Occupancy Debug Chunk</u> | Current Occupancy of the reported structure |
| 000 1000b | <u>Link Debug Request Debug Chunk</u> | Request for link partner to return a NOP.Debug Flit with a specified Debug Chunk Opcode |
| Others | | All other encodings are Reserved |

4.2.3.4.3.2.2 Empty Debug Chunk §

The Empty Debug Chunk is used for padding and alignment. This Debug Chunk does not contain any meaningful debug information and the Debug Chunk Payload (if present) must be ignored by receivers.

The Debug Chunk Continuation bit must be 0. The Debug Chunk Header Size field must be 00b. Receivers must treat any Flit containing Empty Debug Chunks , where these values are not as stated, as a NOP.Empty Flit.

IMPLEMENTATION NOTE: EMPTY DEBUG CHUNK RECOMMENDATIONS §

To reduce parsing effort at the NOP.Debug receiver, it is recommended that consecutive Empty Debug Chunks be avoided. It is more efficient to use a single, larger, Empty Debug Chunk .

There is no constraint on the position of Empty Debug Chunks within a NOP.Debug Flit. Specifically, it is permitted to have an Empty Debug Chunk followed by a non-Empty Debug Chunk.

```
↓↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 64, "isMessage": true, "defaultUnused": "Reserved", "fields": [ {"lsbyte": 0, "lsbit": 1, "msbyte": 0, "msbit": 7, "name": "Empty", "value": ["0", "0", "0", "0", "0", "0", "0"]}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "C", "value": "0"}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 7, "name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "S", "value": ["0", "0"]}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "PCI-SIG Vendor ID", "value": ["0", "0", "0", "0", "0", "0", "0"]}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 7, "msbyte": 4, "msbit": 7, "name": "Empty Debug Chunk Payload (optional, ignored when present)", "attr": "ro"} ]}↓
```

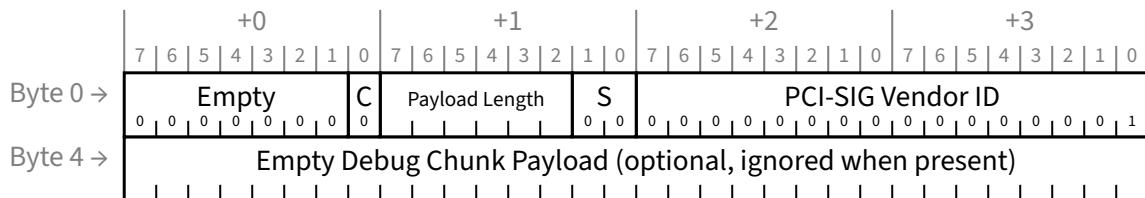


Figure 4-43 Empty Debug Chunk §

4.2.3.4.3.2.3 Start Capture Trigger Debug Chunk §

The Start Capture Trigger Debug Chunk is used to indicate to debug tools such as Logic Analyzers to start capturing what follows the NOP.Debug Flit on the Link.

No Debug Chunk Payload is required for this Debug Chunk, but a transmitter may choose to insert implementation specific content. The Debug Chunk Continuation bit must be 0. Receivers are permitted to silently drop any Start Capture Trigger Debug Chunk, where this value is not as stated.

For ease of use by debug tools, this Debug Chunk is only permitted as the first Debug Chunk of a NOP.Debug Flit.

```
↓↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 64, "isMessage": true, "defaultUnused": "Reserved", "fields": [ {"lsbyte": 0, "lsbit": 1, "msbyte": 0, "msbit": 7, "name": "Start Capture Trigger", "value": ["0", "0", "0", "0", "0", "0", "0", "0"]}, {"addClass": "regFieldSmallText regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "C", "value": "0"}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 7, "name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "S", "value": ["0", "0"]}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "PCI-SIG Vendor ID", "value": ["0", "0", "0", "0", "0", "0", "0"]}, {"addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 7, "msbyte": 4, "msbit": 7, "name": "Debug Chunk Payload (optional, implementation specific)", "attr": "ro"} ]}↓
```

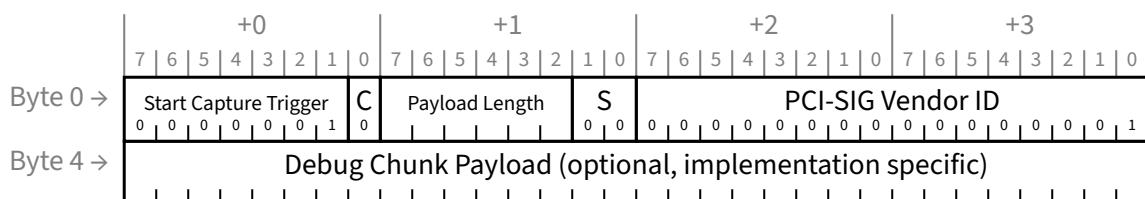


Figure 4-44 Start Capture Trigger Debug Chunk §

4.2.3.4.3.2.4 Stop Capture Trigger Debug Chunk §

The Stop Capture Trigger Debug Chunk is used to indicate to debug tools such as Logic Analyzers to stop capturing what follows the NOP.Debug Flit on the Link.

No Debug Chunk Payload is required for this Debug Chunk but a transmitter may choose to insert implementation specific content. The Continuation bit must be 0. Receivers are permitted to silently drop any Stop Capture Trigger Debug Chunk, where this value is not as stated.

For ease of use by debug tools, this Debug Chunk is only permitted as the first Debug Chunk of a NOP.Debug Flit.

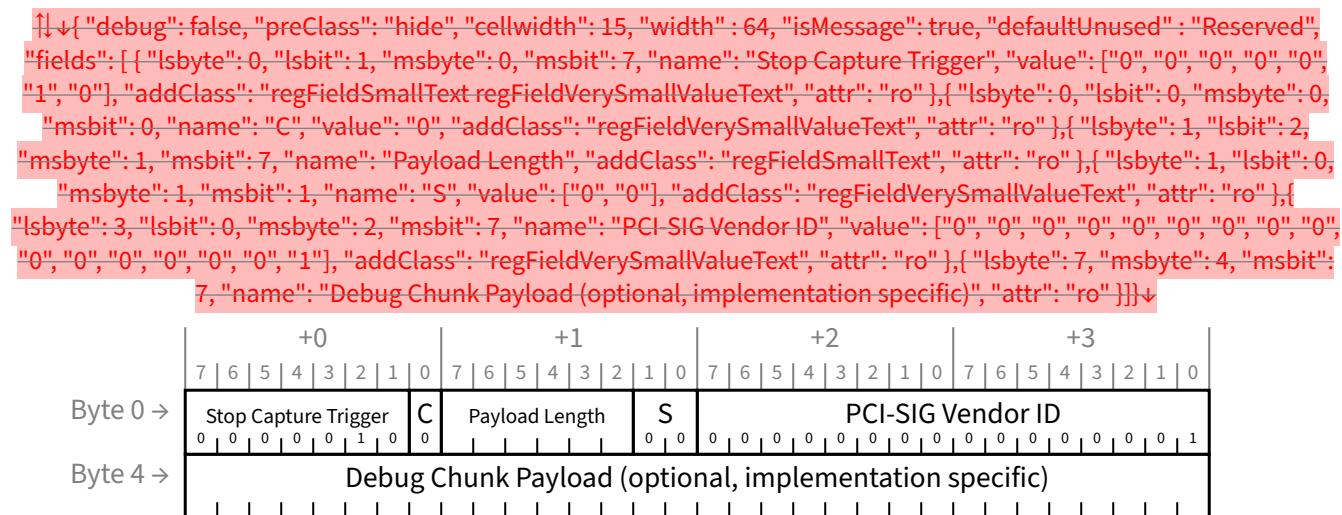


Figure 4-45 Stop Capture Trigger Debug Chunk §

4.2.3.4.3.2.5 FC Information Tracked by Transmitter Debug Chunk §

The FC Information Tracked by Transmitter Debug Chunk transmits the information tracked by a Transmitter for Flow Control TLP Transmission gating.

After the Debug Header, each DW of Debug Chunk Payload consists of a FC Quantity field to indicate the FC quantity information being sent in the remainder of the DW, which contains VC and Header/Data credit values. The Port must only transmit FC quantity information for VCs that it supports.

§ Figure 4-46 displays a Debug Chunk with multiple FC quantities inserted. See § Table 4-26 for the FC Quantity field encodings. The VC field contains the VC number, the Hdr_FC field contains the Header credits for the FC Quantity for that VC, and the Data_FC field contains the Data credits for the FC Quantity for the VC.

Base 6.4 vs Base 6.3

```
    ],{"lsbyte": 0, "msbyte": 0, "name": "FC Info Tracked by Tx", "value": [0, 0, 0, 0, 0, 0, 1, 1], "addClass": "regFieldSmallText regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 0, "msbit": 0, "name": "C", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 7, "name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "S", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "PCI SIG Vendor ID", "value": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], "addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 4, "lsbit": 3, "msbyte": 4, "msbit": 7, "name": "FC Quantity", "attr": "ro"}, {"lsbyte": 4, "lsbit": 0, "msbyte": 4, "msbit": 2, "name": "VC", "attr": "ro"}, {"lsbyte": 5, "msbyte": 5, "msbit": 7, "name": "Hdr_FC", "attr": "ro"}, {"lsbyte": 7, "msbyte": 6, "msbit": 7, "name": "Data_FC", "attr": "ro"}, {"lsbyte": 8, "lsbit": 3, "msbyte": 8, "msbit": 7, "name": "FC Quantity", "attr": "ro"}, {"lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 2, "name": "VC", "attr": "ro"}, {"lsbyte": 9, "msbyte": 9, "msbit": 7, "name": "Hdr_FC", "attr": "ro"}, {"lsbyte": 11, "msbyte": 10, "msbit": 7, "name": "Data_FC", "attr": "ro"}, {"lsbyte": 12, "lsbit": 3, "msbyte": 12, "msbit": 7, "name": "FC Quantity", "attr": "ro"}, {"lsbyte": 12, "lsbit": 0, "msbyte": 12, "msbit": 2, "name": "VC", "attr": "ro"}, {"lsbyte": 13, "msbyte": 13, "msbit": 7, "name": "Hdr_FC", "attr": "ro"}, {"lsbyte": 15, "msbyte": 14, "msbit": 7, "name": "Data_FC", "attr": "ro"}, {"lsbyte": 23, "lsbit": 0, "msbyte": 16, "msbit": 7, "name": "...", "attr": "ro"}]}+}
```

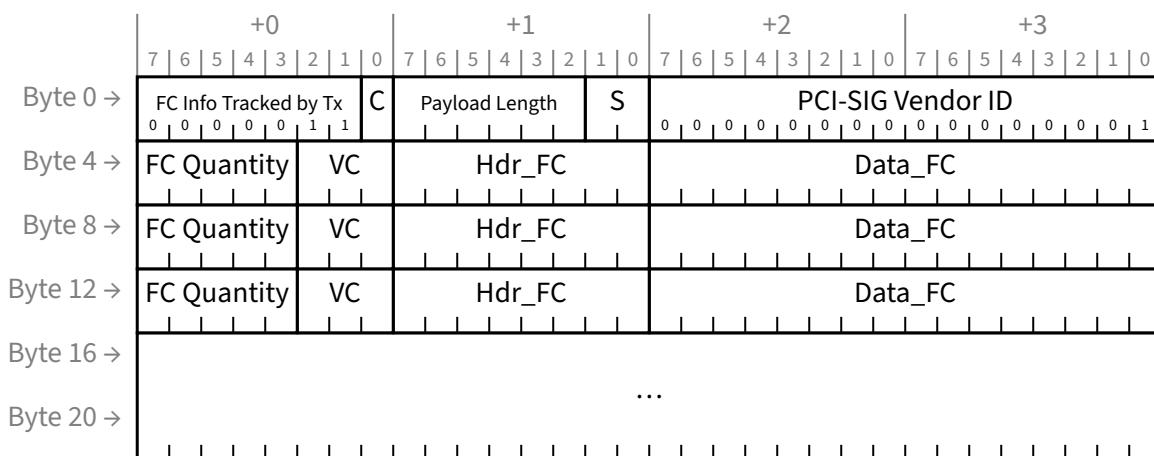


Figure 4-46 FC Information Tracked by Transmitter Debug Chunk §

Table 4-26 FC Information Tracked by Transmitter Encodings

| FC Quantity Encoding | FC Quantity |
|----------------------|-------------------------------------|
| 0 0000b | No valid info |
| 0 0001b | Credit_Consumed_P |
| 0 0010b | Credit_Consumed_NP |
| 0 0011b | Credit_Consumed_CPL |
| 0 0100b | Shared_Credit_Consumed_P |
| 0 0101b | Shared_Credit_Consumed_NP |
| 0 0110b | Shared_Credit_Consumed_CPL |
| 0 0111b | Shared_Credit_Consumed_Currently_P |
| 0 1000b | Shared_Credit_Consumed_Currently_NP |

Base 6.4 vs Base 6.3

| FC Quantity Encoding | FC Quantity |
|----------------------|--|
| 0 1001b | Shared_Credit_Consumed_Currently_CPL |
| 0 1010b | Credit_Limit_P |
| 0 1011b | Credit_Limit_NP |
| 0 1100b | Credit_Limit_CPL |
| 0 1101b | Shared_Credit_Limit_P |
| 0 1110b | Shared_Credit_Limit_NP |
| 0 1111b | Shared_Credit_Limit_CPL |
| 1 0000b | Sum_Shared_Credit_Consumed_P |
| 1 0001b | Sum_Shared_Credit_Consumed_NP |
| 1 0010b | Sum_Shared_Credit_Consumed_CPL |
| 1 0011b | Total_Shared_Credit_Available_P |
| 1 0100b | Total_Shared_Credit_Available_NP |
| 1 0101b | Total_Shared_Credit_Available_CPL |
| 1 0110b | Sum_Shared_Credit_Limit_P |
| 1 0111b | Sum_Shared_Credit_Limit_NP |
| 1 1000b | Sum_Shared_Credit_Limit_CPL |
| 1 1001b | SHARED_CUMULATIVE_CREDITS_REQUIRED_P |
| 1 1010b | SHARED_CUMULATIVE_CREDITS_REQUIRED_NP |
| 1 1011b | SHARED_CUMULATIVE_CREDITS_REQUIRED_CPL |
| 1 1100b | CUMULATIVE_CREDITS_REQUIRED_P |
| 1 1101b | CUMULATIVE_CREDITS_REQUIRED_NP |
| 1 1110b | CUMULATIVE_CREDITS_REQUIRED_CPL |
| 1 1111b | Reserved |

4.2.3.4.3.2.6 FC Information Tracked by Receiver Debug Chunk §

The FC Information Tracked by Receiver Debug Chunk transmits the information tracked by a Receiver for Flow Control TLP Receiver accounting.

After the Debug Chunk Header, each DW of Debug Chunk Payload consists of a FC Quantity field to indicate the FC quantity information being sent in the remainder of the DW, which contains VC and Header/Data credit values. The Port must only transmit FC quantity information for VCs that it supports.

§ Figure 4-47 displays a Debug Chunk with multiple FC quantities inserted. See § Table 4-27 for the FC Quantity field encodings. The VC field contains the VC number, the Hdr_FC field contains the Header credits for the FC Quantity for that VC, and the Data_FC field contains the Data credits for the FC Quantity for the VC.

Base 6.4 vs Base 6.3

```
],{"lsbyte":0,"msbyte":0,"name":"Data_FC","attr":"ro"}, {"lsbyte":0,"msbyte":1,"name":"FC Info Tracked by Rx","value":["0","0","0","0","1","0","0"], "addClass": "regFieldSmallText regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte":0,"msbit":0,"name": "C","attr": "ro"}, {"lsbyte":1,"lsbit":2,"msbyte":1,"msbit":7,"name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte":1,"lsbit":0,"msbyte":1,"msbit":1,"name": "S","attr": "ro"}, {"lsbyte":3,"lsbit":0,"msbyte":2,"msbit":7,"name": "PCI-SIG Vendor ID","value":["0","0","0","0","0","0","0","0"], "addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte":4,"lsbit":3,"msbyte":4,"msbit":7,"name": "FC Quantity", "attr": "ro"}, {"lsbyte":4,"lsbit":0,"msbyte":4,"msbit":2,"name": "VC", "attr": "ro"}, {"lsbyte":5,"msbyte":5,"msbit":7,"name": "Hdr_FC", "attr": "ro"}, {"lsbyte":7,"msbyte":6,"msbit":7,"name": "Data_FC", "attr": "ro"}, {"lsbyte":8,"lsbit":3,"msbyte":8,"msbit":7,"name": "FC Quantity ", "attr": "ro"}, {"lsbyte":8,"lsbit":0,"msbyte":8,"msbit":2,"name": "VC ", "attr": "ro"}, {"lsbyte":9,"msbyte":9,"msbit":7,"name": "Hdr_FC ", "attr": "ro"}, {"lsbyte":11,"msbyte":10,"msbit":7,"name": "Data_FC ", "attr": "ro"}, {"lsbyte":12,"lsbit":3,"msbyte":12,"msbit":7,"name": " FC Quantity ", "attr": "ro"}, {"lsbyte":12,"lsbit":0,"msbyte":12,"msbit":2,"name": "VC ", "attr": "ro"}, {"lsbyte":13,"msbyte":13,"msbit":7,"name": "Hdr_FC ", "attr": "ro"}, {"lsbyte":15,"msbyte":14,"msbit":7,"name": " Data_FC ", "attr": "ro"}, {"lsbyte":23,"lsbit":0,"msbyte":16,"msbit":7,"name": "...", "attr": "ro"}]}+}
```

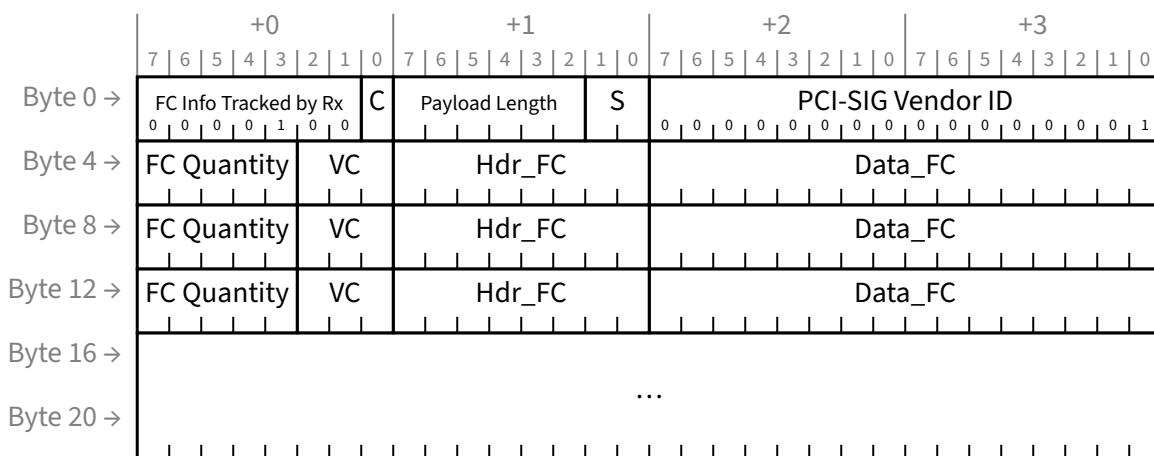


Figure 4-47 \downarrow FC Information Tracked by Receiver Debug Chunk \uparrow \downarrow FC Information Tracked by Receiver Debug Chunk \uparrow

Table 4-27 FC Information Tracked by Receiver Encodings §

| FC Quantity Encoding | FC Quantity |
|----------------------|------------------------------|
| 0 0000b | No valid info |
| 0 0001b | Credits_Allocated_P |
| 0 0010b | Credits_Allocated_NP |
| 0 0011b | Credits_Allocated_CPL |
| 0 0100b | Shared_Credits_Allocated_P |
| 0 0101b | Shared_Credits_Allocated_NP |
| 0 0110b | Shared_Credits_Allocated_CPL |
| 0 0111b | Credits_Received_P |
| 0 1000b | Credits_Received_NP |

| FC Quantity Encoding | FC Quantity |
|----------------------|----------------------------------|
| 0 1001b | Credits_Received_CPL |
| 0 1010b | Shared_Credits_Received_P |
| 0 1011b | Shared_Credits_Received_NP |
| 0 1100b | Shared_Credits_Received_CPL |
| Others | All other encodings are Reserved |

4.2.3.4.3.2.7 Flit Mode Transmitter Retry Flags and Counters Debug Chunk §

The Flit Mode Transmitter Retry Flags and Counters Debug Chunk transmits the transmitter flag and counter values for Flit Sequence Number and retry tracking for Flit Mode.

The Length field must be 2h. Receivers are permitted to silently drop any Flit Mode Transmitter Retry Flags and Counters Debug Chunk , where this value is not as stated.

See § Figure 4-48 for the layout of this Debug Chunk. See § Table 4-28 and § Section 4.2.3.4.2.1 for a description of the fields within this Debug Chunk Payload.

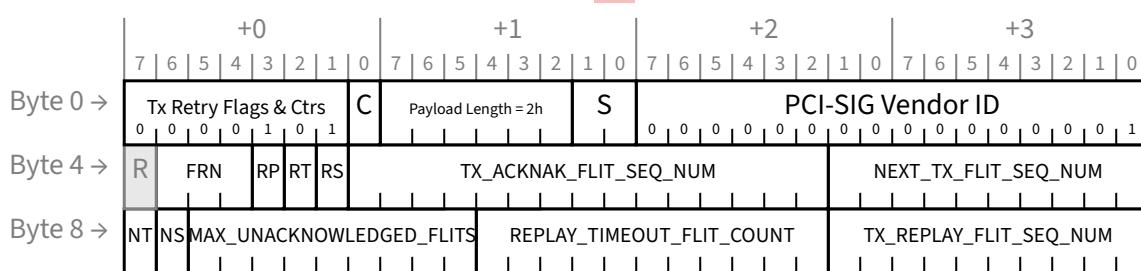


Figure 4-48 Flit Mode Transmitter Retry Flags and Counters Debug Chunk §

Table 4-28 Flit Mode Transmitter Retry Flags and Counters Fields §

| Field | Location |
|--|--|
| Reserved | Byte 4: Bit 7 |
| FLIT_REPLAY_NUM (FRN) | Byte 4: Bits 6:4 |
| REPLAY_IN_PROGRESS (RP) | Byte 4: Bit 3 |
| REPLAY_SCHEDULED_TYPE (RT) | Byte 4: Bit 2 |
| REPLAY_SCHEDULED (RS) | Byte 4: Bit 1 |
| CONSECUTIVE_TX_NAK_FLITS (CN) | { Byte 4: Bit 0, Byte 5: Bits 7:6 } |
| CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLITS (CE) | Byte 5: Bits 5:4 |
| TX_ACKNAK_FLIT_SEQ_NUM | { Byte 5: Bits 3:0, Byte 6: Bits 7:2 } |
| NEXT_TX_FLIT_SEQ_NUM | { Byte 6: Bits 1:0, Byte 7: Bits 7:0 } |

| Field | Location |
|---------------------------|--|
| NAK_SCHEDULED_TYPE (NT) | Byte 8: Bit 7 |
| NAK_SCHEDULED (NS) | Byte 8: Bit 6 |
| MAX_UNACKNOWLEDGED_FLITS | { Byte 8: Bits 5:0, Byte 9: Bits 7:5 } |
| REPLAY_TIMEOUT_FLIT_COUNT | { Byte 9: Bits 4:0, Byte 10: Bits 7:2 } |
| TX_REPLY_FLIT_SEQ_NUM | { Byte 10: Bits 1:0, Byte 11: Bits 7:0 } |

4.2.3.4.3.2.8 Flit Mode Receiver Retry Flags and Counters Debug Chunk §

The Flit Mode Receiver Retry Flags and Counters Debug Chunk transmits the receiver flag and counter values for Flit Sequence Number and retry tracking for Flit Mode.

The Length field must be 2h. Receivers are permitted to silently drop any Flit Mode Receiver Retry Flags and Counters Debug Chunk , where this value is not as stated.

§ Figure 4-49 shows the layout of this Debug Chunk. See § Table 4-29 and § Section 4.2.3.4.2.1 for a description of the fields within the Debug Chunk Payload.

```
↓↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [
    {"lsbyte": 0, "lsbit": 1, "msbyte": 0, "msbit": 7, "name": "Rx Retry Flags & Ctrs", "value": ["0", "0", "0", "0", "0", "1", "1", "0"], "addClass": "regFieldSmallText regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "C", "attr": "ro"}, {"lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 7, "name": "Payload Length = 2h", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "S", "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "PCI-SIG Vendor ID", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "1"], "addClass": "regFieldVerySmallValueText", "attr": "ro"}, {"lsbyte": 4, "lsbit": 6, "msbyte": 4, "msbit": 6, "name": "NI", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 5, "lsbit": 4, "msbyte": 4, "msbit": 5, "name": "ACKD_FLIT_SEQ_NUM", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 6, "lsbit": 2, "msbyte": 2, "msbit": 5, "name": "IMPLICIT_RX_FLIT_SEQ_NUM", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 1, "name": "NEXT_EXPECTED_RX_FLIT_SEQ_NUM", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 8, "lsbit": 7, "msbyte": 8, "msbit": 7, "name": "BO", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 8, "lsbit": 6, "msbyte": 8, "msbit": 6, "name": "WA", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 9, "lsbit": 4, "msbyte": 8, "msbit": 5, "name": "RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 10, "lsbit": 2, "msbyte": 9, "msbit": 3, "name": "NEXT_RX_FLIT_SEQ_NUM_TO_STORE", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 10, "msbit": 1, "name": "NAK_IGNORE_FLIT_SEQ_NUM", "addClass": "regFieldVerySmallText", "attr": "ro"}]}↓
```

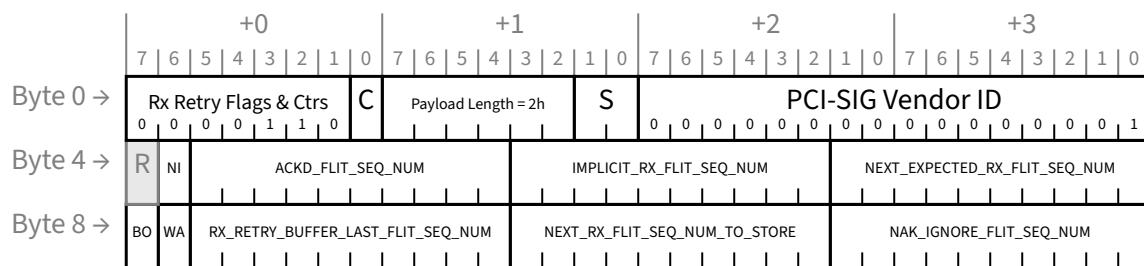


Figure 4-49 Flit Mode Receiver Retry Flags and Counters Debug Chunk §

Table 4-29 Flit Mode Receiver Retry Flags and Counters Fields §

| Field | Location |
|--|--|
| Reserved | Byte 4: Bit 7 |
| NON_IDLE_EXPLICIT_SEQ_NUM_FLIT_RCVD (NI) | Byte 4: Bit 6 |
| ACKD_FLIT_SEQ_NUM | { Byte 4: Bits 5:0, Byte 5: Bits 7:4 } |
| IMPLICIT_RX_FLIT_SEQ_NUM | { Byte 5: Bits 3:0, Byte 6: Bits 7:2 } |
| NEXT_EXPECTED_RX_FLIT_SEQ_NUM | { Byte 6: Bits 1:0, Byte 7: Bits 7:0 } |
| RX_RETRY_BUFFER_OVERFLOW (BO) | Byte 8: Bit 7 |
| NAK_WITHDRAWAL_ALLOWED (WA) | Byte 8: Bit 6 |
| RX_RETRY_BUFFER_LAST_FLIT_SEQ_NUM | { Byte 8: Bits 5:0, Byte 9: Bits 7:4 } |
| NEXT_RX_FLIT_SEQ_NUM_TO_STORE | { Byte 9: Bits 3:0, Byte 10: Bits 7:2 } |
| NAK_IGNORE_FLIT_SEQ_NUM | { Byte 10: Bits 1:0, Byte 11: Bits 7:0 } |

4.2.3.4.3.2.9 Buffer Occupancy Debug Chunk §

The Buffer Occupancy Debug Chunk transmits the current occupancy of the reported structure.

After the Debug Chunk Header, each DW of Debug chunk Payload consists of a Buffer ID field to indicate the buffer structure being sent in the remainder of the DW, which contains Number of Occupied Entries values. § Figure 4-50 shows a Buffer Occupancy Debug Chunk with multiple buffers being reported. See § Table 4-30 for the Buffer ID field encodings. The Number of Occupied Entries field contains the current occupancy value for the Buffer ID.

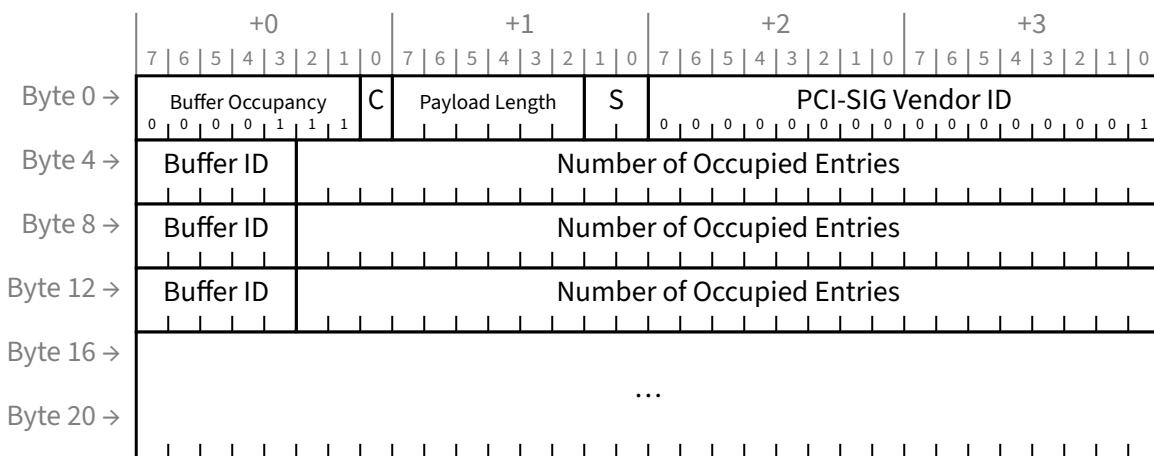


Figure 4-50 Buffer Occupancy Debug Chunk §

Table 4-30 Buffer Occupancy Encodings §

| Buffer ID Encoding | Buffer ID |
|--------------------|----------------------------------|
| 0h | TX Retry Buffer |
| 1h | RX Retry Buffer |
| Others | All other encodings are Reserved |

4.2.3.4.3.2.10 Link Debug Request Debug Chunk §

The Link Debug Request Debug Chunk requests the Receiver to return a NOP.Debug Flit with the requested Debug Chunk Opcode.

§ Figure 4-51 shows the layout of the Link Debug Request Debug Chunk . After the Debug Chunk Header, a single DW of Debug Chunk Payload contains the Requested Debug Chunk Opcode and Requested Debug Chunk Vendor ID.

For ease of use by the Receiver, this Debug Chunk is only permitted as the first chunk of the NOP.Debug Flit. Receiver responds on a best effort basis, and is permitted to ignore the request if it cannot service it.

```
↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 64, "isMessage": true, "defaultUnused": "Reserved", "fields": [ { "lsbyte": 0, "lsbit": 1, "msbyte": 0, "msbit": 7, "name": "Link Debug Request", "value": ["0", "0", "0", "1", "0", "0", "0"], "addClass": "regFieldSmallText regFieldVerySmallValueText", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 0, "name": "C", "attr": "ro" }, { "lsbyte": 1, "lsbit": 2, "msbyte": 1, "msbit": 7, "name": "Payload Length", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 1, "name": "S", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "PCI SIG Vendor ID", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "1"], "addClass": "regFieldVerySmallValueText", "attr": "ro" }, { "lsbyte": 4, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Req. Debug Chunk Op.", "addClass": "regFieldSmallText", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "Requested Debug Chunk Vendor ID", "addClass": "regFieldSmallText", "attr": "ro" } ] } ↓
```

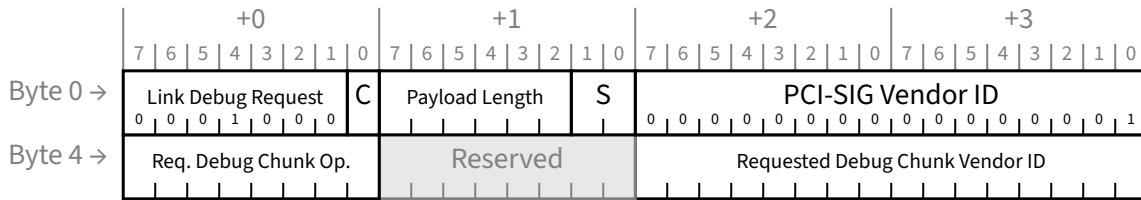


Figure 4-51 Link Debug Request Debug Chunk

4.2.3.4.3.3 NOP.Vendor Flit

A NOP.Vendor Flit (see § Figure 4-52) is a general purpose type that can be utilized by a vendor to transmit proprietary information. After the NOP Flit Common Header, the next DW contains a Vendor ID field that identifies the vendor associated with this Flit, and all other bytes in the Flit are available for vendor usage. It is recommended for vendors to include a sub-type field.

```
↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 128, "isMessage": true, "defaultUnused": "Reserved",  
      "fields": [ { "lsbyte": 0, "lsbit": 4, "msbyte": 0, "msbit": 7, "name": "NOPVendor (Fh)", "addClass": "",  
      "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 1, "msbyte": 0, "msbit": 3, "name": "NOP Flit Counter", "attr": "ro" }, {  
      "lsbyte": 3, "msbyte": 3, "msbit": 7, "name": "NOP Stream ID", "attr": "ro" }, { "lsbyte": 5, "msbyte": 4, "msbit": 7, "name": "  
      "Content defined by Vendor", "attr": "ro" }, { "lsbyte": 7, "msbyte": 6, "msbit": 7, "name": "Vendor ID", "attr": "ro" }, {  
      "lsbyte": 15, "msbyte": 8, "msbit": 7, "name": "228 bytes of content defined by Vendor", "attr": "ro" } ] } ↓
```

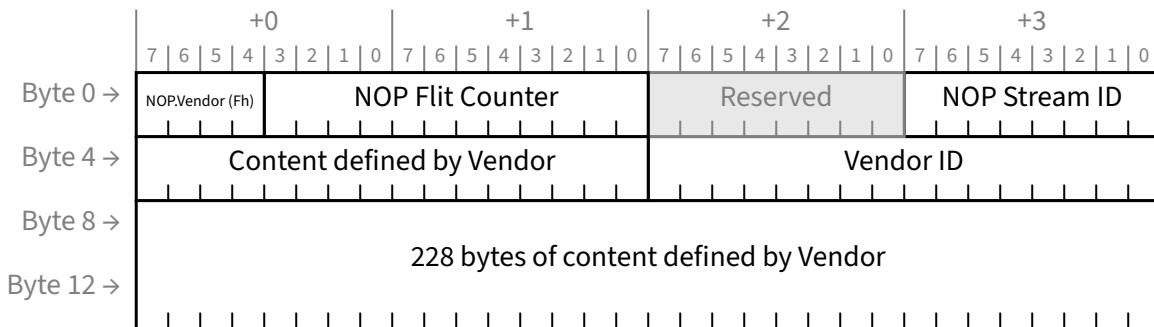
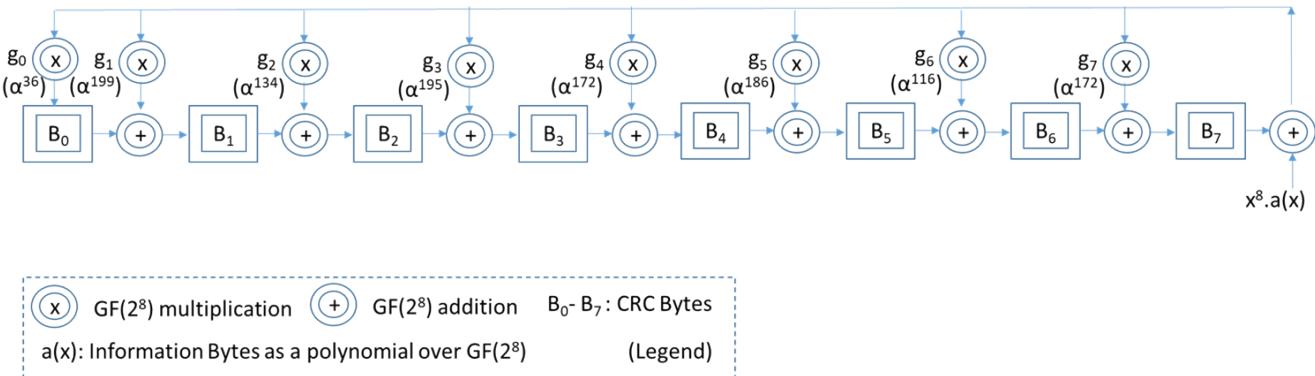


Figure 4-52 NOP.Vendor Flit Payload

4.2.3.4.4 CRC Bytes in Flit §

The CRC generator polynomial, defined over $\text{GF}(2^8)$, is $g(x) = (x+\alpha)(x+\alpha^2)\dots(x+\alpha^8)$, where α is the root of the primitive polynomial of degree 8: $x^8 + x^5 + x^3 + x + 1$. Thus, $g(x) = x^8 + \alpha^{172}x^7 + \alpha^{116}x^6 + \alpha^{186}x^5 + \alpha^{172}x^4 + \alpha^{195}x^3 + \alpha^{134}x^2 + \alpha^{199}x + \alpha^{36}$. § Figure 4-53 demonstrates how CRC bytes are generated for the Transmit as well as Receive side. On the Receive side the generated CRC bytes are compared against the received CRC bytes (after the FEC decode/ correct) and any mismatch represents an uncorrectable error.



Bytes 0-241 form the input $a(x)$ whereas B_0-B_7 represent CRC0-crc7

Figure 4-53 CRC generation/ checking in Flit §

The CRC generator using MATLAB code to output the generator matrix, the generator matrix, as well as the RTL code is provided in § Appendix K. . The CRC bytes are generated by 242 Bytes of Data * Gen Matrix. Thus, Data is 1x1936 bits, Gen matrix is 1936x64 bits. Data is arranged from MSB to LSB – [(241,7), (241,6), (241,5), (241,4) (0,7), (0,6),(0,5),(0,4),(0,3),(0,2),(0,1),(0,0)] where (x,y) is yth bit of xth byte. The generator matrix appears in § Appendix K. and must be considered the standard to implement the CRC during encode and decode. Any pipelined implementation must match this mechanism. The code is provided for the Transmit side. It takes 242 Bytes as input and generates the 8B of CRC. On the Receive side, after the ECC decode/ correction, the first 242 Bytes of the Flit will be used to generate the expected 8 Bytes of CRC and each of these 8 Bytes will be compared against the received 8B of CRC. Any mismatch results in the Flit being declared not Valid and a replay requested, if needed, as described earlier.

4.2.3.4.5 ECC Bytes in Flit §

The FEC code is a 3-way interleaved, as described before, where the ECC Symbols are defined over $\text{GF}(2^8)$. Thus, three consecutive Bytes in a wire belong to three different ECC groups, as shown by different colors in § Table 4-10 through § Table 4-14 . Thus, any burst of length ≤ 16 in a wire is guaranteed not to impact more than one Symbol in a code word which is capable of correcting single byte errors in an effort to minimize the FEC latency.

A sample RTL code for the FEC is provided in § Appendix J. . The rest of this section describes the construction of the FEC code. The sample RTL in the appendix is the reference should any confusion arise out of the description here. While an implementation may follow a different approach (e.g., pipelining), it must still match the RTL sample results for all permutations of values in the bytes both for the encoder as well as the decoder function.

Let α be the root of a primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$ (represented as 0x1D). Hence, $\alpha^8 = \alpha^4 + \alpha^3 + \alpha^2 + 1$ which can also be represented as {0001_1101}. Thus, each 8-bit symbol (over GF(2⁸)) can be expressed as a polynomial as powers of alpha (1 through 254) or as a degree 8 polynomial. We calculate powers of α along with an 8-bit representation as follows: Should be: α^0 (=1): {0000_0001}, α^1 : {0000_0010}, ..., α^7 : {1000_0000}, α^8 : {0001_1101},... α^{255} = 1. Thus, we get 255 distinct numbers as powers of α (all 0s is not a power of α), as shown in § Figure 4-54 . The log function is given in § Figure 4-55 (which is basically the inverse of constructing the power of α from a given 8-bit non-zero Syndrome which will be used for error correction). The H matrix consists of two parts: Horizontal Parity and Check bits, as shown in § Figure 4-56 and § Figure 4-57 . The Horizontal Parity is the bit-wise XORs of Symbols (see § Equation 4-1). The Check bits (see § Equation 4-2) is similar to a CRC calculation represented as follows:

$$\uparrow\downarrow P = \sum_{i=0}^{83} B_i$$

Equation 4-1 Parity bytes §

$$P = \sum_{i=0}^{83} B_i$$

$$\uparrow\downarrow C = \sum_{i=0}^{83} B_i \times \alpha^{(84-i)}$$

Equation 4-2 Check bytes §

$$C = \sum_{i=0}^{83} B_i \times \alpha^{(84-i)}$$

Base 6.4 vs Base 6.3

α is the root of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$

$i \rightarrow \alpha^i$ (in hex)

| |
|---|
| 00: 01 01: 02 02: 04 03: 08 04: 10 05: 20 06: 40 07: 80 08: 1d 09: 3a 0a: 74 0b: e8 0c: cd 0d: 87 0e: 13 0f: 26 |
| 10: 4c 11: 98 12: 2d 13: 5a 14: b4 15: 75 16: ea 17: c9 18: 8f 19: 03 1a: 06 1b: 0c 1c: 18 1d: 30 1e: 60 1f: c0 20: 9d 21: 27 22: 4e |
| 23: 9c 24: 25 25: 4a 26: 94 27: 35 28: 6a 29: d4 2a: b5 2b: 77 2c: ee 2d: c1 2e: 9f 2f: 23 30: 46 31: 8c 32: 05 33: 0a 34: 14 35: 28 |
| 36: 50 37: a0 38: 5d 39: ba 3a: 69 3b: d2 3c: b9 3d: 6f 3e: de 3f: a1 40: 5f 41: be 42: 61 43: c2 44: 99 45: 2f 46: 5e 47: bc 48: 65 |
| 49: ca 4a: 89 4b: 0f 4c: 1e 4d: 3c 4e: 78 4f: f0 50: fd 51: e7 52: d3 53: bb 54: 6b 55: d6 56: b1 57: 7f 58: fe 59: e1 5a: df 5b: a3 |
| 5c: 5b 5d: b6 5e: 71 5f: e2 60: d9 61: af 62: 43 63: 86 64: 11 65: 22 66: 44 67: 88 68: 0d 69: 1a 6a: 34 6b: 68 6c: d0 6d: bd 6e: |
| 67 6f: ce 70: 81 71: 1f 72: 3e 73: 7c 74: f8 75: ed 76: c7 77: 93 78: 3b 79: 76 7a: ec 7b: c5 7c: 97 7d: 33 7e: 66 7f: cc 80: 85 81: |
| 17 82: 2e 83: 5c 84: b8 85: 6d 86: da 87: a9 88: 4f 89: 9e 8a: 21 8b: 42 8c: 84 8d: 15 8e: 2a 8f: 54 90: a8 91: 4d 92: 9a 93: 29 94: |
| 52 95: a4 96: 55 97: aa 98: 49 99: 92 9a: 39 9b: 72 9c: e4 9d: d5 9e: b7 9f: 73 a0: e6 a1: d1 a2: bf a3: 63 a4: c6 a5: 91 a6: 3f a7: |
| 7e a8: fc a9: e5 aa: d7 ab: b3 ac: 7b ad: f6 ae: f1 af: ff b0: c3 b1: db b2: ab b3: 4b b4: 96 b5: 31 b6: 62 b7: c4 b8: 95 b9: 37 ba: |
| 6e bb: dc bc: a5 bd: 57 be: ae bf: 41 c0: 82 c1: 19 c2: 32 c3: 64 c4: c8 c5: 8d c6: 07 c7: 0e c8: 1c c9: 38 ca: 70 cb: e0 cc: dd cd: |
| a7 ce: 53 cf: a6 d0: 51 d1: a2 d2: 59 d3: b2 d4: 79 d5: f2 d6: f9 d7: ef d8: c3 d9: 9b da: 2b db: 56 dc: ac dd: 45 de: 8a df: 09 e0: |
| 12 e1: 24 e2: 48 e3: 90 e4: 3d e5: 7a e6: f4 e7: f5 e8: f7 e9: f3 ea: fb eb: eb ec: cb ed: 8b ee: 0b ef: 16 f0: 2c f1: 58 f2: b0 f3: 7d |
| f4: fa f5: e9 f6: cf f7: 83 f8: 1b f9: 36 fa: 6c fb: d8 fc: ad fd: 47 fe: 8e ff: 01 |

| |
|---|
| 00: 01 01: 02 02: 04 03: 08 04: 10 05: 20 06: 40 07: 80 |
| 08: 1d 09: 3a 0a: 74 0b: e8 0c: cd 0d: 87 0e: 13 0f: 26 |
| 10: 4c 11: 98 12: 2d 13: 5a 14: b4 15: 75 16: ea 17: c9 |
| 18: 8f 19: 03 1a: 06 1b: 0c 1c: 18 1d: 30 1e: 60 1f: c0 |
| 20: 9d 21: 27 22: 4e 23: 9c 24: 25 25: 4a 26: 94 27: 35 |
| 28: 6a 29: d4 2a: b5 2b: 77 2c: ee 2d: c1 2e: 9f 2f: 23 |
| 30: 46 31: 8c 32: 05 33: 0a 34: 14 35: 28 36: 50 37: a0 |
| 38: 5d 39: ba 3a: 69 3b: d2 3c: b9 3d: 6f 3e: de 3f: a1 |
| 40: 5f 41: be 42: 61 43: c2 44: 99 45: 2f 46: 5e 47: bc |
| 48: 65 49: ca 4a: 89 4b: 0f 4c: 1e 4d: 3c 4e: 78 4f: f0 |
| 50: fd 51: e7 52: d3 53: bb 54: 6b 55: d6 56: b1 57: 7f |
| 58: fe 59: e1 5a: df 5b: a3 5c: 5b 5d: b6 5e: 71 5f: e2 |
| 60: d9 61: af 62: 43 63: 86 64: 11 65: 22 66: 44 67: 88 |
| 68: 0d 69: 1a 6a: 34 6b: 68 6c: d0 6d: bd 6e: 67 6f: ce |
| 70: 81 71: 1f 72: 3e 73: 7c 74: f8 75: ed 76: c7 77: 93 |
| 78: 3b 79: 76 7a: ec 7b: c5 7c: 97 7d: 33 7e: 66 7f: cc |
| 80: 85 81: 17 82: 2e 83: 5c 84: b8 85: 6d 86: da 87: a9 |
| 88: 4f 89: 9e 8a: 21 8b: 42 8c: 84 8d: 15 8e: 2a 8f: 54 |
| 90: a8 91: 4d 92: 9a 93: 29 94: 52 95: a4 96: 55 97: aa |
| 98: 49 99: 92 9a: 39 9b: 72 9c: e4 9d: d5 9e: b7 9f: 73 |
| a0: e6 a1: d1 a2: bf a3: 63 a4: c6 a5: 91 a6: 3f a7: 7e |
| a8: fc a9: e5 aa: d7 ab: b3 ac: 7b ad: f6 ae: f1 af: ff |
| b0: e3 b1: db b2: ab b3: 4b b4: 96 b5: 31 b6: 62 b7: c4 |
| b8: 95 b9: 37 ba: 6e bb: dc bc: a5 bd: 57 be: ae bf: 41 |
| c0: 82 c1: 19 c2: 32 c3: 64 c4: c8 c5: 8d c6: 07 c7: 0e |
| c8: 1c c9: 38 ca: 70 cb: e0 cc: dd cd: a7 ce: 53 cf: a6 |
| d0: 51 d1: a2 d2: 59 d3: b2 d4: 79 d5: f2 d6: f9 d7: ef |
| d8: c3 d9: 9b da: 2b db: 56 dc: ac dd: 45 de: 8a df: 09 |
| e0: 12 e1: 24 e2: 48 e3: 90 e4: 3d e5: 7a e6: f4 e7: f5 |
| e8: f7 e9: f3 ea: fb eb: eb ec: cb ed: 8b ee: 0b ef: 16 |
| f0: 2c f1: 58 f2: b0 f3: 7d f4: fa f5: e9 f6: cf f7: 83 |
| f8: 1b f9: 36 fa: 6c fb: d8 fc: ad fd: 47 fe: 8e ff: 01 |

Figure 4-54 FEC Table: i to α^i

Base 6.4 vs Base 6.3

α is the root of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$

$a^i \rightarrow i$ (in hex) ~~11 00: ff 01: 00 02: 01 03: 19 04: 02 05: 32 06: 1a 07: c6 08: 03 09: df 0a: 33 0b: ee 0c: 1b 0d: 68 0e: c7 0f: 4b 10: 04 11: 64 12: e0 13: 0e 14: 34 15: 8d 16: e0 17: 81 18: 1c 19: c1 1a: 69 1b: f8 1c: c8 1d: 08 1e: 4c 1f: 71 20: 05 21: 8a 22: 65 23: 2f 24: e1 25: 24 26: 0f 27: 21 28: 35 29: 93 2a: 8e 2b: da 2c: f0 2d: 12 2e: 82 2f: 45 30: 1d 31: b5 32: c2 33: 7d 34: 6a 35: 27 36: f9 37: b9 38: c9 39: 9a 3a: 09 3b: 78 3c: 4d 3d: e4 3e: 72 3f: a6 40: 06 41: bf 42: 8b 43: 62 44: 66 45: dd 46: 30 47: fd 48: e2 49: 98 4a: 25 4b: b3 4c: 10 4d: 91 4e: 22 4f: 88 50: 36 51: d0 52: 94 53: ce 54: 8f 55: 96 56: db 57: bd 58: f1 59: d2 5a: 13 5b: 5c 5c: 83 5d: 38 5e: 46 5f: 40 60: 1e 61: 42 62: b6 63: a3 64: c3 65: 48 66: 7e 67: 6e 68: 6b 69: 3a 6a: 28 6b: 54 6c: fa 6d: 85 6e: ba 6f: 3d 70: ca 71: 5e 72: 9b 73: 9f 74: 0a 75: 15 76: 79 77: 2b 78: 4e 79: d4 7a: e5 7b: ac 7c: 73 7d: f3 7e: a7 7f: 57 80: 07 81: 70 82: c0 83: f7 84: 8c 85: 80 86: 63 87: 0d 88: 67 89: 4a 8a: de 8b: ed 8c: 31 8d: c5 8e: fe 8f: 18 90: e3 91: a5 92: 99 93: 77 94: 26 95: b8 96: b4 97: 7c 98: 11 99: 44 9a: 92 9b: d9 9c: 23 9d: 20 9e: 89 9f: 2e a0: 37 a1: 3f a2: d1 a3: 5b a4: 95 a5: bc a6: cf a7: cd a8: 90 a9: 87 aa: 97 ab: b2 ac: dc ad: fc ae: be af: 61 b0: f2 b1: 56 b2: d3 b3: ab b4: 14 b5: 2a b6: 5d b7: 9e b8: 84 b9: 3c ba: 39 bb: 53 bc: 47 bd: 6d be: 41 bf: a2 c0: 1f c1: 2d c2: 43 c3: d8 c4: b7 c5: 7b c6: a4 c7: 76 c8: c4 c9: 17 ca: 49 cb: ec cc: 7f cd: 0c ce: 6f cf: f6 d0: 6c d1: a1 d2: 3b d3: 52 d4: 29 d5: 9d d6: 55 d7: aa d8: fb d9: 60 da: 86 db: b1 dc: bb dd: cc de: 3e df: 5a e0: cb e1: 59 e2: 5f e3: b0 e4: 9c e5: a9 e6: a0 e7: 51 e8: 0b e9: f5 ea: 16 eb: eb ec: 7a ed: 75 ee: 2c ef: d7 f0: 4f f1: ae f2: d5 f3: e9 f4: e6 f5: e7 f6: ad f7: e8 f8: 74 f9: d6 fa: f4 fb: ea fc: a8 fd: 50 fe: 58 ff: af~~

00: ff 01: 00 02: 01 03: 19 04: 02 05: 32 06: 1a 07: c6
08: 03 09: df 0a: 33 0b: ee 0c: 1b 0d: 68 0e: c7 0f: 4b
10: 04 11: 64 12: e0 13: 0e 14: 34 15: 8d 16: ef 17: 81
18: 1c 19: c1 1a: 69 1b: f8 1c: c8 1d: 08 1e: 4c 1f: 71
20: 05 21: 8a 22: 65 23: 2f 24: e1 25: 24 26: 0f 27: 21
28: 35 29: 93 2a: 8e 2b: da 2c: f0 2d: 12 2e: 82 2f: 45
30: 1d 31: b5 32: c2 33: 7d 34: 6a 35: 27 36: f9 37: b9
38: c9 39: 9a 3a: 09 3b: 78 3c: 4d 3d: e4 3e: 72 3f: a6
40: 06 41: bf 42: 8b 43: 62 44: 66 45: dd 46: 30 47: fd
48: e2 49: 98 4a: 25 4b: b3 4c: 10 4d: 91 4e: 22 4f: 88
50: 36 51: d0 52: 94 53: ce 54: 8f 55: 96 56: db 57: bd
58: f1 59: d2 5a: 13 5b: 5c 5c: 83 5d: 38 5e: 46 5f: 40
60: 1e 61: 42 62: b6 63: a3 64: c3 65: 48 66: 7e 67: 6e
68: 6b 69: 3a 6a: 28 6b: 54 6c: fa 6d: 85 6e: ba 6f: 3d
70: ca 71: 5e 72: 9b 73: 9f 74: 0a 75: 15 76: 79 77: 2b
78: 4e 79: d4 7a: e5 7b: ac 7c: 73 7d: f3 7e: a7 7f: 57
80: 07 81: 70 82: c0 83: f7 84: 8c 85: 80 86: 63 87: 0d
88: 67 89: 4a 8a: de 8b: ed 8c: 31 8d: c5 8e: fe 8f: 18
90: e3 91: a5 92: 99 93: 77 94: 26 95: b8 96: b4 97: 7c
98: 11 99: 44 9a: 92 9b: d9 9c: 23 9d: 20 9e: 89 9f: 2e
a0: 37 a1: 3f a2: d1 a3: 5b a4: 95 a5: bc a6: cf a7: cd
a8: 90 a9: 87 aa: 97 ab: b2 ac: dc ad: fc ae: be af: 61
b0: f2 b1: 56 b2: d3 b3: ab b4: 14 b5: 2a b6: 5d b7: 9e
b8: 84 b9: 3c ba: 39 bb: 53 bc: 47 bd: 6d be: 41 bf: a2
c0: 1f c1: 2d c2: 43 c3: d8 c4: b7 c5: 7b c6: a4 c7: 76
c8: c4 c9: 17 ca: 49 cb: ec cc: 7f cd: 0c ce: 6f cf: f6
d0: 6c d1: a1 d2: 3b d3: 52 d4: 29 d5: 9d d6: 55 d7: aa
d8: fb d9: 60 da: 86 db: b1 dc: bb dd: cc de: 3e df: 5a
e0: cb e1: 59 e2: 5f e3: b0 e4: 9c e5: a9 e6: a0 e7: 51
e8: 0b e9: f5 ea: 16 eb: eb ec: 7a ed: 75 ee: 2c ef: d7
f0: 4f f1: ae f2: d5 f3: e9 f4: e6 f5: e7 f6: ad f7: e8
f8: 74 f9: d6 fa: f4 fb: ea fc: a8 fd: 50 fe: 58 ff: af

Figure 4-55 FEC Log Table: a^i to $i \oplus j$

Base 6.4 vs Base 6.3

↑↓H = 1 1 ... 1 0 1 α 84 α 83 ... α 1 0 Row Parity Check Bits MathType@MTEF@5@5@+=
 feaahqart1ev3aaatCvAUfeBSjuyZL2yd9gzLbvyNv2CaerbuLwBLn
 hiov2DGi1BTfMBaeXatLxBI9gBaerb9wDYLwzYbItLDharqqtubsr
 4rNCHbGeaGqiVu0Je9sqqrpepC0xbbL8F4rqqFFfpeea0xe9Lq=Jc9
 vqaqpepm0xbba9pwe9Q8fs0=yqaqpepae9pg0FirpepeKkFr0xfr=x
 fr=xb9adbaqaaeGaciGaaiaabeqaaamaabaabaaGcbaGaamisaiabg2
 da9maaemaabaqbamqabiGbaaaabaGaaGymaaqaaiaaigdaaaeacaGG
 UaGaaOlaiac6caaeeacaalXaaabaGaaGimaaqaaiaaigdaaaeacaq
 aHXoqydaahaaWcbeqaaiaaildacaal0aaaaaGcbaGaeqySde2aaWba
 aSqabeaacaal4aGaaG4maaaaaOqaaiaac6cacaGGUaGaaOlaaqaaai
 abeg7aHbqaaiaaigdaaaeacaalWaay5bSlaawla7aabaaaWx
 biMapeqbamqabiqaaaqaaiaabkfacaqGVbGaaeiDaiaabccacaqGqb
 GaaeyyaiaabkhacaqGPbGaaeiDaiaabMhaaaacaqGdbGaaeiAaiaa
 bwgacaqGjbGaae4AaiaabccacaqGcbGaaeyAaiaabshacaqGZbaaaa aa@5F27@↓

$$H = \left| \begin{array}{cccccc} 1 & 1 & \dots & 1 & 0 & 1 \\ \alpha^{84} & \alpha^{83} & \dots & \alpha & 1 & 0 \end{array} \right| \begin{array}{l} \text{Row Parity} \\ \text{Check Bits} \end{array}$$

Figure 4-56 H-matrix of the FEC §

Base 6.4 vs Base 6.3

| | B_0 | \cdots | B_{82} | B_{83} | B_{84} | B_{85} |
|---|---------------|----------|------------|------------|------------|----------|
| 0 | α^{84} | \cdots | α^2 | α | 1 | p_0 |
| 1 | α^{85} | \cdots | α^3 | α^2 | α | p_1 |
| 2 | α^{86} | \cdots | α^4 | α^3 | α^2 | p_2 |
| 3 | α^{87} | \cdots | α^5 | α^4 | α^3 | p_3 |
| 4 | α^{88} | \cdots | α^6 | α^5 | α^4 | p_4 |
| 5 | α^{89} | \cdots | α^7 | α^6 | α^5 | p_5 |
| 6 | α^{90} | \cdots | α^8 | α^7 | α^6 | p_6 |
| 7 | α^{91} | \cdots | α^9 | α^8 | α^7 | p_7 |

Check Bits

Row Parity [N-3:0]

Powers of alpha for the check bits for Bytes 0 to 84

Figure 4-57 Weight of check bits for different Bytes/bits §

The Transmitter has to calculate B_{N-1} (horizontal parity) and B_{N-2} (Check Symbol) during encoding from the remaining $(N-2)$ Bytes, as described above. The same mechanism will be used to compute the corresponding two Syndrome bytes at the Receiver. $N = 86$. However, only one of the three groups has 86B in its code word, whereas the other two have 85 Bytes each to form the 256B Flit. This is resolved by forcing $B_{83} = 0$ for the information part so that two groups that have 85 Bytes each effectively become 86B with 84B as information (the last Byte being 0) and two Bytes of ECC. The mapping of the Flit Byte i to an ECC group and the Byte-offset within the group can be derived as follows:

↑↑ For $0 \leq i \leq 249$, Byte i in Flit is mapped to ECC Group $i \bmod 3$ with Byte Offset $\lfloor i/3 \rfloor$ ECC Group 1 Byte Offset 83 is 0
 ECC Group 2 Byte Offset 83 is 0 For $250 \leq i \leq 255$ (the ECC Bytes), Byte i in Flit is mapped to ECC Group $(i-249) \bmod 3$ with Byte Offset $\lceil i/3 \rceil$ ↓

For $0 \leq i \leq 249$,
 Byte i in Flit is mapped to ECC Group $i \bmod 3$
 with Byte Offset $\text{floor}(i/3)$
 ECC Group 1 Byte Offset 83 is 0
 ECC Group 2 Byte Offset 83 is 0
 For $250 \leq i \leq 255$ (the ECC Bytes),
 Byte i in Flit is mapped to ECC Group $(i-249) \bmod 3$
 with Byte Offset $\text{ceil}(i/3)$

Thus:

- Byte 0 of Flit maps to ECC Group 0, Byte offset 0;
- Byte 1 of Flit maps to ECC Group 1, Byte offset 0;
- Byte 2 of Flit maps to ECC group 2, Byte offset 0;
- Byte 3 of Flit maps to ECC Group 0, Byte offset 1;
- Byte 4 of Flit maps to ECC Group 1, Byte offset 1;
- ...
- Byte 248 maps to ECC group 2, Byte Offset 82;
- Byte 249 maps to ECC group 0, Byte offset 83.
- ECC group 1, Byte offset 83 is 00h.
- ECC group 2, Byte offset 83 is 00h.

The ECC Bytes are then allocated as:

- Bytes 250 and 253 of the Flit maps to ECC Group 1, Byte offsets 84 and 85 respectively;
- Bytes 251 and 254 of the Flit maps to ECC Group 2, Byte offsets 84 and 85 respectively; and
- Bytes 252 and 255 of the Flit maps to ECC Group 0, Bytes offsets 84 and 85 respectively.

The basic idea is that horizontal parity (Synd_Parity) identifies the bits in the Symbol (or Byte) that has flipped and the check bits (Synd_Check) identifies the Symbol number that has the error. Looking at the expanded bit arrangement, one can see that knowing the bit positions one can reverse map the column number. § Figure 4-58 shows the ECC decode function. The Syndrome Parity is the XOR of $B_0 \dots B_{83}$ and B_{85} (Synd_Parity). The Syndrome Check (Synd_Check) is calculated by computing the expected B_{84} from $B_0 \dots B_{83}$ received and XORing with received B_{84} . Once the two Syndrome Symbols (Bytes) are calculated, the following steps are taken:

| Synd_Check | Synd_Parity | Description |
|------------|-------------|---|
| = 00h | = 00h | No error |
| = 00h | ≠ 00h | Error in B_{85} : Corrected $B_{85} = B_{85} \wedge \text{Synd_Parity}$ |
| ≠ 00h | = 00h | Error in B_{84} : Corrected $B_{84} = B_{84} \wedge \text{Synd_Check}$ |
| ≠ 00h | ≠ 00h | Follow the log table for Synd_Check, Synd_Parity and then follow the modulo subtraction +1 logic to identify the failing column number. It is possible that we point to a non-existing column, in that case it is an uncorrectable error. Once the column number is known, we can correct the Symbol and the corresponding CRC correction (if applicable) |

Base 6.4 vs Base 6.3

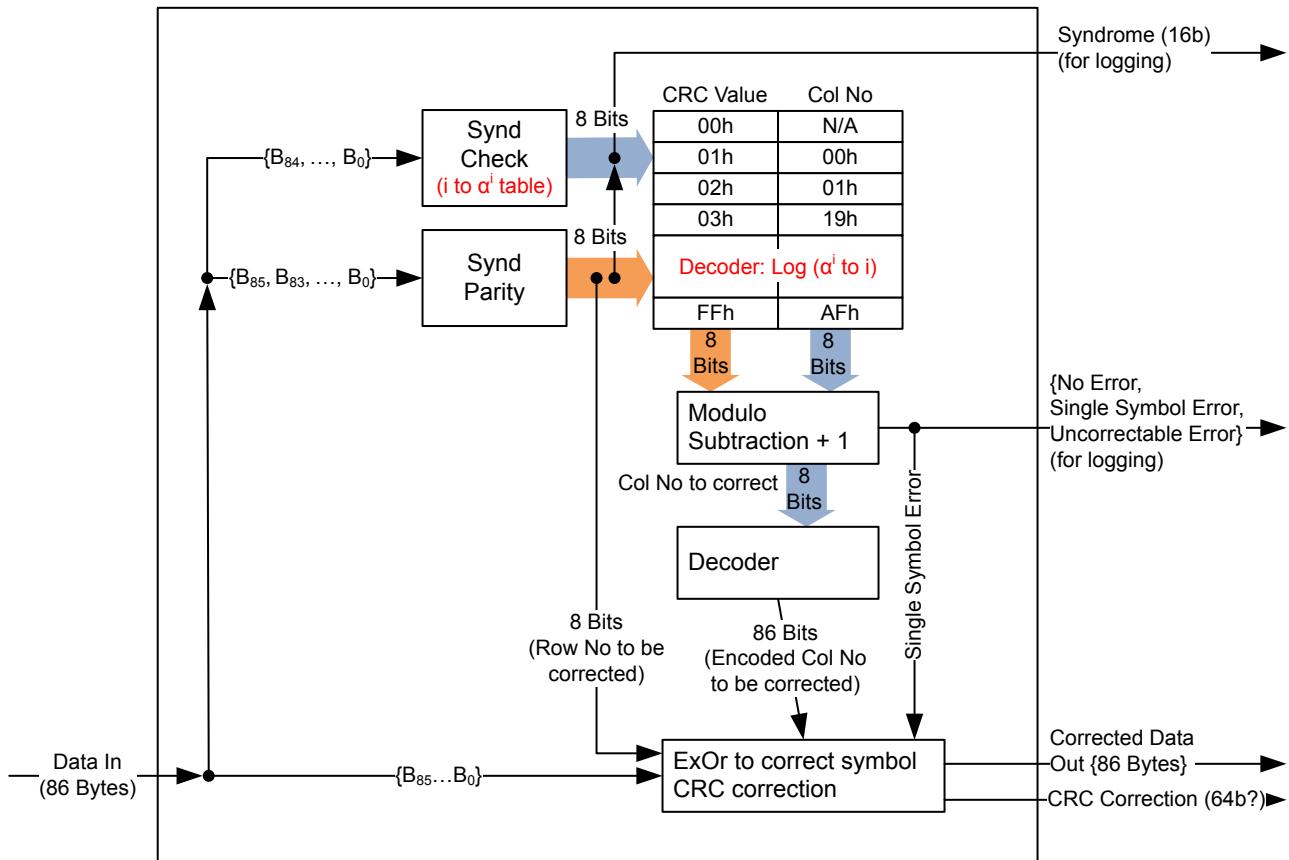


Figure 4-58 ECC Decoder function §

The Receive side check is as follows on the 256B flit (as shown in § Figure 4-59). Each of the three ECC decoders, performs correction and error reporting as needed. In the final stage of CRC check, a decision is made whether the received flit can be accepted or retried.

Base 6.4 vs Base 6.3

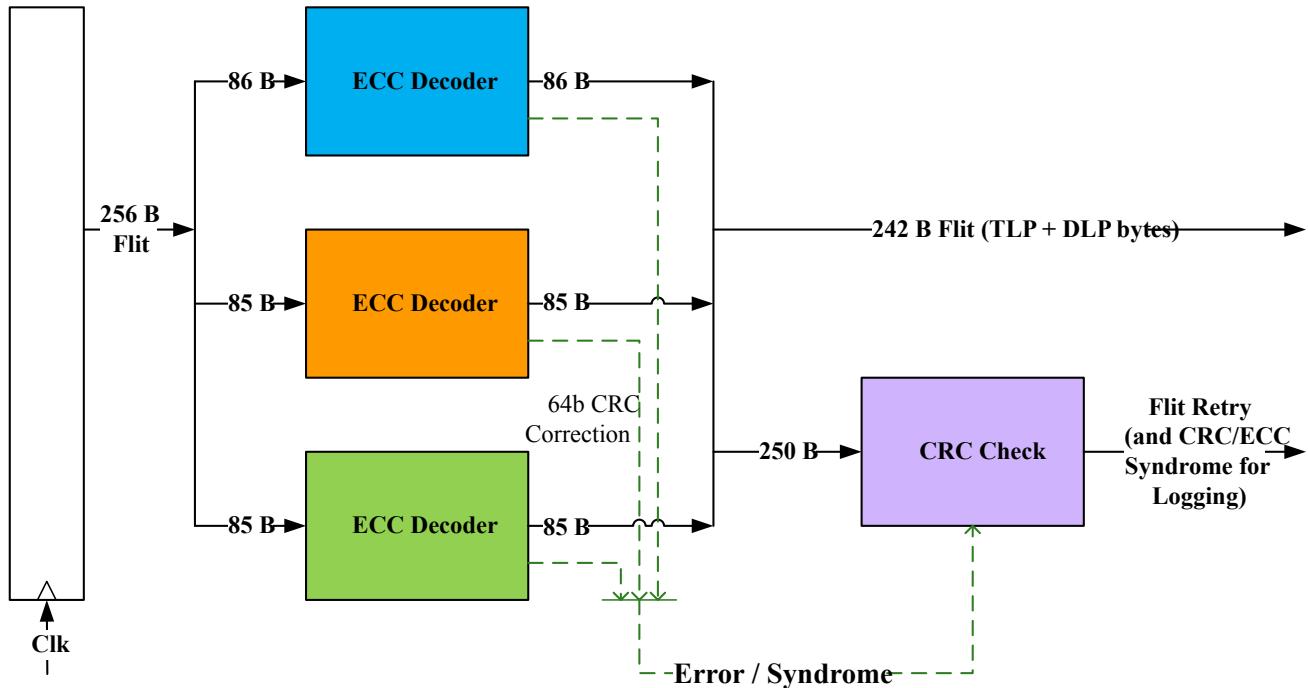


Figure 4-59 3-way ECC decode followed by CRC check of flit on the Receive side §

4.2.3.4.6 Ordered Set insertion in Data Stream in Flit Mode §

For Data Rates of 2.5 GT/s or 5.0 GT/s, the SKP Ordered Set insertion interval does not change (e.g., between 1180 and 1538 Symbols), but occurs at the Flit boundary. The other Ordered Sets that can occur with up-size or down-size with L0p is provided in § Section 4.2.6.7.

For Data Rates of 8.0 GT/s and above, the following rules apply:

- A Control SKP Ordered Set must immediately follow the SDS Ordered Set sequence marking the start of the Data Stream, irrespective of when the prior Control SKP Ordered Set was transmitted.
- Once the Data Stream has started, an Ordered Set must be transmitted in fixed intervals, as defined § Table 4-32, as follows:
 - If the Data Stream will continue, a SKP Ordered Set must be transmitted on all the active (or to be activated) Lanes.
 - On an up-size with L0p : The Lanes that will be activated to join the active Lanes, an SDS Ordered Set must be transmitted on all the Lanes that will be activated (not in those that are already active) just prior to sending the SKP Ordered Set across all the wider Link, after which Flits will be sent on the wider Link
 - Else If the Link will enter a low-power state, an EIOSQ must be transmitted on all the active Lanes
 - Else (the Link needs to enter Recovery) : An EIEOS must be transmitted on all the configured Lanes of the Link
 - If the Link enters Recovery , the appropriate SKP Ordered Set must be transmitted immediately after the first Ordered Set (EIEOS).

Table 4-32 Ordered Set insertion interval once Data Stream starts in terms of number of Flits §

| Link Width and Clocking Modes | x16 | x8 | x4 | x2 | x1 |
|--|-----|-----|-----|----|----|
| Common Clock/SRNS Mode with 1b/1b encoding | 748 | 374 | 187 | 93 | 46 |
| Common Clock/SRNS Mode with 128b/130b encoding | 374 | 187 | 93 | 46 | 23 |
| SRIS Mode with 1b/1b encoding | 74 | 37 | 18 | 9 | 4 |
| SRIS Mode with 128b/130b encoding | 37 | 18 | 9 | 4 | 2 |

IMPLEMENTATION NOTE: CONSECUTIVE SKP ORDERED SETS IN FLIT MODE §

Two consecutive SKP Ordered Sets in a Data Stream in Flit Mode are always equidistant for a given width and clocking mode when either 1b/1b encoding or 128b/130b encoding is used. For example, for a x1 Link in SRIS Mode with 128b/130b encoding, a SKP Ordered Set is always sent after two Flits, which will occupy 32 Data Blocks. However, the same spacing cannot be guaranteed with 8b/10b encoding, as the SKPs are “scheduled” every 1180 to 1583 Symbol times in non-SRIS mode and less than 154 Symbol times in SRIS mode. For example, a x1 Link in SRIS mode with 8b/10b encoding sends an average of 1.66 (=256/154) SKP Ordered Sets after every Flit (256 Symbols): sometimes it sends one SKP Ordered Set after every Flit and other times it sends two back-to-back SKP Ordered Sets.

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

The Link equalization procedure enables components to adjust the Transmitter and the Receiver setup of each Lane to improve the signal quality and meet the requirements specified in § Chapter 8., when operating at 8.0 GT/s and higher data rates. All the Lanes that are associated with the LTSSM (i.e., those Lanes that are currently operational or may be operational in the future due to Link Upconfigure) must participate in the equalization procedure. The procedure must be executed during the first data rate change to any data rate at 8.0 GT/s or above, unless all components in the Link have advertised that no equalization is needed. Components must arrive at the appropriate Transmitter setup for all the operating conditions and data rates that they will encounter in the future when LinkUp =1b. Components must not require that the equalization procedure be repeated at any data rate for reliable operation, although there is provision to repeat the procedure. Components must store the Transmitter setups that were agreed to during the equalization procedures and use them for future operation at 8.0 GT/s and higher data rates. Components are permitted to fine-tune their Receiver setup even after the equalization procedure is complete as long as doing so does not cause the Link to be unreliable (i.e., does not meet the requirements in § Chapter 8.) or go to Recovery .

The Link equalization procedure is not required for any data rates and can be completely bypassed if all components in the Link have advertised that no equalization is needed in its TS1 / TS2 Ordered Sets or Modified TS1/TS2 Ordered Sets (see § Table 4-36 , § Table 4-37 , and § Table 4-38). A component may choose to advertise that it does not need equalization at any rates above 5.0 GT/s if it supports 32.0 GT/s or higher data rates and can either operate reliably with equalization settings stored from a prior equalization procedure or does not need equalization for reliable operation.

The equalization procedure can be initiated either autonomously or by software. It is strongly recommended that components use the autonomous mechanism for all the data rates above 5.0 GT/s that they intend to operate in. However, a component that chooses not to participate in the autonomous mechanism for all the data rates above 5.0 GT/s must have its associated software ensure that the software based mechanism is applied to the data rates above 5.0 GT/s where the autonomous mechanism was not applied, prior to operating at that data rate.

Normally, equalization is performed at a higher data rate only if equalization has successfully completed at all lower data rates above 5.0 GT/s. For example, a Link will complete equalization successfully at 8.0 GT/s, followed by 16.0 GT/s, followed by 32.0 GT/s, followed by 64.0 GT/s. However, an optional mechanism to begin the equalization procedures at the highest NRZ rate supported, 32.0 GT/s, is permitted if all components support data rates of 32.0 GT/s or higher and the mechanism is supported by all components in the Link, as advertised in the TS1 / TS2 Ordered sets or Modified TS1 / TS2 Ordered Sets. When this optional mechanism is enabled and successfully negotiated between the components, equalization is not performed the 8.0 GT/s and 16.0 GT/s data rates; the equalization procedures are performed at all common data rates above 16.0 GT/s beginning with the 32.0 GT/s data rate. For all the data rates above 5.0 GT/s where equalization is not performed (specifically 8.0 GT/s and 16.0 GT/s), the expectation is that the Link will not operate at those data rates and the components will not advertise those data rates (neither 8.0 GT/s nor 16.0 GT/s) as the highest data rate supported. For example, a Link may train to L0 in 2.5 GT/s, enter Recovery and perform equalization at 32.0 GT/s, and then re-enter Recovery and perform equalization at 64.0 GT/s, skipping equalization at 8.0 GT/s and 16.0 GT/s. In this case, the intended data rates of operation of the Link are 2.5 GT/s, 5.0 GT/s, 32.0 GT/s, or 64.0 GT/s. If any of the equalization procedures attempted at 32.0 GT/s or 64.0 GT/s is unsuccessful even after re-equalization attempts and the Link needs to equalize at lower data rates, the Downstream Port must stop advertising Equalization Bypass to Highest NRZ Rate Support and ensure that the Link returns to operation at 2.5 GT/s or 5.0 GT/s. The required equalization procedures are then performed as they would have been if the optional mechanism to skip over equalization to the highest common data rate(s) was never supported. If the equalization procedure at the highest NRZ rate supported is initially successful but the Link is not able to operate reliably at that data rate, the Downstream Port is permitted to initiate a speed change to any lower data rate followed by an equalization procedure at that rate (when necessary). The requirements of § Table 4-33 must be followed. The Upstream Port must disable Equalization Bypass to Highest NRZ Rate support as soon as an equalization procedure at the 8.0 GT/s data rate is performed. If the equalization procedure at the lower data rates is driven by software, it must set the Equalization Bypass to Highest NRZ Rate Disable and No Equalization Needed Disable register bits to 1b each; set the target Link speed such that the Link will be operational at 2.5 GT/s or 5.0 GT/s; and then set the target Link speed to equalize at the lower rates starting with 8.0 GT/s onwards. A port must not advertise the Equalization Bypass to Highest NRZ Rate Support if the Equalization Bypass to Highest NRZ Rate Disable bit is Set to 1b.

Another optional mechanism to skip the entire equalization process and go directly to the highest common data rate is permitted if all components support data rates of 32.0 GT/s or higher and the No Equalization Needed mechanism is supported by all components in the Link, as advertised in the TS1 / TS2 or Modified TS1/TS2 Ordered sets. This is done if a component is either able to retrieve the equalization and other circuit settings at all the data rates from a prior equalization that will work for the component, or it does not need equalization at all in all data rates above 5.0 GT/s. A component must not advertise this capability if the Equalization Bypass to Highest NRZ Rate Disable bit is Set to 1b. Components supporting 64.0 GT/s or higher data rate are strongly encouraged to support this mode, especially after the equalization has been performed at least once since power was applied. This would be helpful for faster resume times after the devices in the Link go through deep power savings states.

If one direction of the Link is advertising No Equalization Needed and the other side is advertising Equalization Bypass to Highest NRZ Rate Support in the TS1 / TS2 Ordered Sets, the Link will operate in the Equalization Bypass to Highest NRZ Rate Support since the No Equalization Needed bit also indicates that the device is capable of bypassing Equalization to the highest data rate. In the Modified TS1/TS2 Ordered Sets, a device that sets No Equalization Needed bit to 1b must also set the Equalization Bypass to Highest NRZ Rate Support to 1b. If one direction of the Link is advertising No Equalization Needed bit and the other side is advertising Equalization Bypass to Highest NRZ Rate Support only in the Modified TS1/TS2 Ordered Sets, the Link will operate in the Equalization Bypass to Highest NRZ Rate Support. Link operation is undefined if a device advertises No Equalization Needed bit as 1b and Equalization Bypass to Highest NRZ Rate Support to 0b in the Modified TS1/TS2 Ordered Sets it transmits. Components are permitted to go straight to 64.0 GT/s from any data rate without having to go through 32.0 GT/s data rate if No Equalization Needed has been negotiated, since no equalization is involved.

The autonomous mechanism is executed if both components advertise that they are capable of at least the 8.0 GT/s data rate (via the TS1 and TS2 Ordered Sets) during the initial Link negotiation (when LinkUp is set to 1b) and the Downstream Port chooses to perform the equalization procedure at the intended data rates of operation above 5.0 GT/s. While not recommended, the Downstream Port may choose to perform the autonomous mechanism only on a subset of the

intended data rates of operation above 5.0 GT/s. In that case, the software based mechanism must be executed in order to perform the equalization procedure for the intended data rates of operation above 5.0 GT/s, not covered by the autonomous mechanism. For example, if both components advertised 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s Data Rates but autonomous equalization was performed for only 8.0 GT/s and 16.0 GT/s Data Rates, then software based mechanism must be adopted for equalization at 32.0 GT/s Data Rate.

In the autonomous mechanism, after entering L0, irrespective of the current Link speed, neither component must transmit any DLLP, except the NOP2 DLLP, if the equalization procedure must be performed and until the equalization procedure completes. The equalization procedure is considered complete once the Transmitter and Receiver setup of each Lane has been adjusted for each common data rate supported above 5.0 GT/s for which the Downstream Port intends to perform equalization using the autonomous mechanism. The Downstream Port is required to initiate the speed change to the data rate where the equalization needs to be performed. During any equalization (autonomous or software initiated or re-equalization), the Downstream Port must not advertise support for any data rate above the data rate for which equalization needs to be performed in Recovery. The following example is provided to illustrate the equalization flow.

Example: Consider a Link where equalization needs to be performed autonomously at 8.0 GT/s, and 16.0 GT/s. The Downstream Port enters Recovery to perform equalization at 8.0 GT/s by not advertising any data rates above 8.0 GT/s. The 8.0 GT/s equalization procedure is deemed to have been successfully executed if the Equalization 8.0 GT/s Phase 3 Successful bit and Equalization 8.0 GT/s Complete bit of the Link Status 2 register are both set to 1b. Immediately following the transition from Recovery to L0, after the initial data rate change to 8.0 GT/s, the Downstream Port is required to transition from L0 to Recovery, advertise 16.0 GT/s data rate support (but not advertise support for 32.0 GT/s, even if it is capable of supporting 32.0 GT/s), change the data rate to 16.0 GT/s and perform the 16.0 GT/s equalization procedure.

If the Downstream Port detects equalization problems or the Upstream Port made an equalization redo request (by setting the Request Equalization bit to 1b) the Downstream Port may redo equalization prior to proceeding to operate at the data rate where the equalization failed or performing equalization at a higher data rate. The number of back-to-back equalization redos at a given data rate is implementation specific but must be finite. If at the conclusion of the initial or subsequent equalization process and the execution of an implementation specific number of equalization redo's, the Link is not able to operate reliably at the data rate where equalization was performed, then it must revert back to a lower data rate of operation.

Components using the autonomous mechanism must not initiate any autonomous Link width downsizing until the equalization procedure completes. An Upstream Port must not transmit any DLLP, except the NOP2 DLLP, until it receives a non-NOP2 DLLP from the Downstream Port. If the Downstream Port performs equalization again, it must not transmit any DLLP, except the NOP2 DLLP, until it completes the equalization procedure. A Downstream Port may perform equalization again based on its own needs or based on the request from the Upstream Port, if it can meet its system requirements. Executing equalization multiple times may interfere with software determination of Link and device status, as described in § Section 6.6.

IMPLEMENTATION NOTE: DLLP BLOCKING DURING AUTONOMOUS EQUALIZATION §

When using the autonomous mechanism for equalization at 8.0 GT/s or higher data rates, the Downstream Port is required to block the transmission of DLLPs until equalization has completed at all data rates for which the autonomous equalization mechanism will be performed, and the Upstream Port is required to block the transmission of DLLPs until a DLLP is received from the Downstream Port. If both components advertise that they are capable of the 16.0 GT/s (or higher) data rate but the Downstream Port only uses the autonomous mechanism for equalization at 8.0 GT/s, the Downstream Port is only required to block DLLP transmission until 8.0 GT/s equalization has completed. Similarly, if both components advertise that they are capable of the 32.0 GT/s data rate but the Downstream Port only uses the autonomous mechanism for equalization at 16.0 GT/s, the Downstream Port is only required to block DLLP transmission until 16.0 GT/s equalization has completed. If the Downstream Port delays entering Recovery from L0 while DLLP transmission is blocked, either the L0 Inferred Electrical Idle timeout (see § [Section 4.2.5.4](#)) or the DLLP timeout (see § [Section 2.6.1.2](#)) may expire in the Upstream or Downstream Ports. If either of these two timeouts occurs, it will result in the initiation of an entry to Recovery to perform Link retraining. Neither of these two timeouts is a reportable error condition, and the resulting Link retraining has no impact on proper Link operation.

The DLLP limitations described in this implementation note do not apply to NOP2 DLLPs. When operating in Flit Mode, since entry to Recovery must happen on an Ordered Set boundary, Flit transmission may occur prior to the completion of equalization at all data rates for which the autonomous equalization mechanism will be performed. These Flits will be transmitting NOP2 DLLPs. The receipt of NOP2 DLLPs cannot be used by the Upstream Port to unblock the transmission of non-NOP2 DLLPs.

When using the software based mechanism, software must guarantee that there will be no side-effects for transactions in flight (e.g., no timeout), if any, due to the Link undergoing the equalization procedure. Software can write 1b to the Perform Equalization bit in the Link Control 3 Register, followed by a write to the Target Link Speed field in the Link Control 2 register to enable the Link to run at 8.0 GT/s or higher, followed by a write of 1b to the Retrain Link bit in the Link Control register of the Downstream Port to perform equalization. Software must not enable the Link to run at a data rate above 8.0 GT/s during a software initiated equalization procedure if the equalization procedure at all the lower data rates starting with 8.0 GT/s have not been successfully executed and the Link is not capable of bypassing equalization to higher data rate(s) (i.e., either Equalization Bypass to Highest NRZ Rate Supported is 0b or Equalization Bypass to Highest NRZ Rate Disable is 1b). The equalization procedure is deemed successful as follows for the following data rates:

- 8.0 GT/s: Equalization 8.0 GT/s Phase 3 Successful bit and Equalization 8.0 GT/s Complete bit of the Link Status 2 register are both set to 1b
- 16.0 GT/s: Equalization 16.0 GT/s Phase 3 Successful bit and Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register are both set to 1b
- 32.0 GT/s: Equalization 32.0 GT/s Phase 3 Successful bit and Equalization 32.0 GT/s Complete bit of the [↑↓32.0 GT/s Link Status Register](#) [↑↓32.0 GT/s Status Register](#) are both set to 1b
- 64.0 GT/s: Equalization 64.0 GT/s Phase 3 Successful bit and Equalization 64.0 GT/s Complete bit of the [↑↓64.0 GT/s Link Status Register](#) [↑↓64.0 GT/s Status Register](#) are both set to 1b

Software may set the Hardware Autonomous Width Disable bit of the Link Control register in both components, or use some other mechanism to ensure that the Link is in its full functional width prior to setting the Perform Equalization bit in the Downstream Port. The component that had initiated the autonomous width downsizing is responsible to upconfigure the Link to go to its full functional width by initiating the transition to Recovery and Configuration within 1 ms of the Hardware Autonomous Width Disable bit being set to 1b. If an Upstream Port does not advertise the 8.0 GT/s data rate, the 16.0 GT/s data rate, the 32.0 GT/s, or the 64.0 GT/s data rate initially and did not participate in the autonomous

equalization mechanism for the non-advertised rates, its associated software must ensure there will be no side-effects for transactions in flight, if any, during equalization, before it instructs the Upstream Port to go to Recovery and advertise the previously non-advertised data rates and initiate a speed change. The Downstream Port subsequently initiates the equalization procedure during the initial speed change to the data rate advertised by the Upstream Port when it transitions to Recovery .

Upstream Ports are required to check for equalization setting problems in the Recovery.RcvrLock state (see § Section 4.2.7.4.1). However, both Downstream and Upstream Ports are permitted to use implementation specific methods to detect equalization problems at any time. A Port that detects a problem with its equalization settings is required to undertake the following actions, for each the following data rates:

- 8.0 GT/s: Link Equalization Request 8.0 GT/s bit in the Link Status 2 Register is set to 1b.
- 16.0 GT/s: Link Equalization Request 16.0 GT/s bit in the 16.0 GT/s Status Register is set to 1b.
- 32.0 GT/s: Link Equalization Request 32.0 GT/s bit in the 32.0 GT/s Status Register is set to 1b.
- 64.0 GT/s: Link Equalization Request 64.0 GT/s bit in the 64.0 GT/s Status Register is set to 1b.

In addition to setting the appropriate Link Equalization Request bit to 1b, an Upstream Port must initiate a transition to Recovery (if necessary) and request equalization at the appropriate data rate in the Recovery.RcvrCfg state by setting the Request Equalization bit of its transmitted TS2 Ordered Sets to 1b and the Equalization Request Data Rate bits to the data rate of the detected problem. If it requests equalization, it must request equalization for each detected problem only once. When requesting equalization, the Upstream Port is also permitted, but not required, to set the Quiesce Guarantee bit to 1b to inform the Downstream Port that an equalization process initiated within 1 ms will not cause any side-effects to its operation.

When a Downstream Port receives an equalization request from an Upstream Port (when it is in the Recovery.RcvrCfg state and receives 8 consecutive TS2 Ordered Sets with the Request Equalization bit set to 1b), it must either initiate an equalization process at the requested data rate (as defined by the received Equalization Request Data Rate bits) within 1 ms of completing the next Recovery to L0 transition, or it must set the appropriate Link Equalization Request 8.0 GT/s in its Link Status 2 register, Link Equalization Request 16.0 GT/s bit in its 16.0 GT/s Status Register , Link Equalization Request 32.0 GT/s bit in its 32.0 GT/s Status Register , or Link Equalization Request 64.0 GT/s bit in its 64.0 GT/s Status Register . It should initiate an equalization process only if it can guarantee that executing the equalization process will not cause any side-effects to either its operation or the Upstream Port's operation. The Downstream Port is permitted, but not required, to use the received Quiesce Guarantee bit to determine the Upstream Port's ability to execute an equalization process without side-effects.

If a Downstream Port wants to initiate an equalization process and can guarantee that it will not cause side-effects to its own operation but is unable to directly determine whether the equalization process will cause side-effects to the Upstream Port's operation, then it is permitted to request that the Upstream Port initiate an equalization request. The Downstream Port does so by transitioning to Recovery and in the Recovery.RcvrCfg state setting the Request Equalization bit of its transmitted TS2 Ordered Sets to 1b, the Equalization Request Data Rate bits to the desired data rate, and the Quiesce Guarantee bit to 1b. When an Upstream Port receives such an equalization request from a Downstream Port (when it is in the Recovery.RcvrCfg state and receives 8 consecutive TS2 Ordered Sets with the Request Equalization and Quiesce Guarantee bits set to 1b), it is permitted, but not required, to quiesce its operation and prepare to execute an equalization process at the data rate requested by the Downstream Port, and then request equalization at that same data rate (using the method described previously for reporting equalization setting problems) and with the Quiesce Guarantee bit set to 1b. There is no time limit on how long the Upstream Port can take to respond, but it should attempt to do so as quickly as possible. If a Downstream Port makes a request and receives such a response from the Upstream Port, then it must either initiate an equalization process at the agreed-upon data rate within 1 ms of completing the next Recovery to L0 transition if it can still guarantee that executing the equalization process will not cause any side-effects to its operation, or it must set the appropriate Link Equalization Request 8.0 GT/s bit in its Link Status 2 register, Link Equalization Request 16.0 GT/s bit in its 16.0 GT/s Status Register , Link Equalization Request 32.0 GT/s bit in its 32.0 GT/s Status Register , or Link Equalization Request 64.0 GT/s bit in its 64.0 GT/s Status Register .

IMPLEMENTATION NOTE: USING QUIESCE GUARANTEE MECHANISM §

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

A component may operate at a lower data rate after reporting its equalization problems, either by timing out through Recovery.Speed or by initiating a data rate change to a lower data rate. Any data rate change required to perform the equalization procedure is exempt from the 200 ms requirement in § Section 6.11 . § Table 4-33 describes the mechanism for performing redo Equalization. Sometimes it may be necessary to perform a speed change to an intermediate data rate to redo equalization. For example, if the Downstream Port wants to redo equalization at 16.0 GT/s, bypass equalization is not supported, and the current data rate is either 2.5 GT/s or 5.0 GT/s, the Downstream Port must first initiate a speed change to 8.0 GT/s (the 8.0 GT/s equalization procedure will not be executed unless necessary) from which it will launch the redo equalization for 16.0 GT/s. The equalization procedure can be performed at most once in each trip through the Recovery state.

Table 4-33 Equalization Requirements Under Different Conditions §

| | |
|---|--|
| From 2.5 GT/s or 5.0 GT/s to 8.0 GT/s Equalization | <p>The mechanisms described here are identical for all flavors of equalization: initial or redo equalization; autonomous or software initiated.</p> <p>The Downstream Port communicates the Transmitter preset values and the Receiver preset hints, if applicable, for each Lane to the Upstream Port using 8b/10b encoding. These values are communicated using the EQ TS2 Ordered Sets (defined in § Section 4.2.5.1) in Recovery.RcvrCfg , when a data rate change to the higher data rate has been negotiated, prior to transitioning to the higher data rate to perform equalization. The preset values sent in the EQ TS2 Ordered Sets are derived as follows:</p> <ul style="list-style-type: none"> • For equalization at 8.0 GT/s: Upstream Port 8.0 GT/s Transmitter Preset and Upstream Port 8.0 GT/s Receiver Preset Hint fields of each Lane Equalization Control Register Entry . <p>After the data rate change to the higher data rate where equalization needs to be performed, the Upstream Port transmits TS1 Ordered Sets with the preset values it received. The preset values must be within the operable range defined in § Section 8.3.3.3 if reduced swing will be used by the Transmitter.</p> <p>After the data rate change to the higher data rate where equalization needs to be performed, the Downstream Port transmits TS1 Ordered Sets with the preset values as follows with the assumption that the preset values must be within the operable range defined in § Section 8.3.3.3 if reduced swing will be used by the Transmitter:</p> <ul style="list-style-type: none"> • For equalization at 8.0 GT/s: Downstream Port 8.0 GT/s Transmitter Preset and optionally Downstream Port 8.0 GT/s Receiver Preset Hint fields of each Lane Equalization Control Register Entry . <p>To perform redo equalization, the Downstream Port must request speed change through EQ TS1 Ordered Sets in Recovery.RcvrLock at 2.5 GT/s or 5.0 GT/s to inform the Upstream Port that it intends to redo equalization at the higher data rate. An Upstream Port should advertise the higher data rate in Recovery if it receives EQ TS1 Ordered Sets with speed change bit set to 1b and if it intends to operate at the higher data rate in the future.</p> |
| From 8.0 GT/s to 16.0 GT/s Equalization, from 16.0 GT/s to | <p>The mechanisms described here are identical for all flavors of equalization: initial or redo equalization; autonomous or software initiated.</p> <p>When negotiating to the higher data rate, the Downstream Port communicates the Transmitter preset values for each Lane to the Upstream Port using 128b/130b encoding. These values are communicated using 128b/130b EQ TS2 Ordered Sets (defined in § Section 4.2.5.1) in Recovery.RcvrCfg , when a data rate change to the higher</p> |

Base 6.4 vs Base 6.3

| | |
|--|---|
| <p>32.0 GT/s Equalization, OR from 32.0 GT/s to 64.0 GT/s Equalization</p> | <p>data rate has been negotiated, prior to transitioning to the higher data rate. The preset values sent in the <u>128b/130b EQ TS2 Ordered Sets</u> are derived as follows:</p> <ul style="list-style-type: none"> For equalization at 16.0 GT/s: <u>Upstream Port 16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. For equalization at 32.0 GT/s: <u>Upstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. For equalization at 64.0 GT/s: <u>Upstream Port 64.0 GT/s Transmitter Preset field of the 64.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. <p> Optionally, the Upstream Port communicates initial Transmitter preset settings to the Downstream Port using the <u>128b/130b EQ TS2 Ordered Sets</u> sent in <u>Recovery.RcvrCfg</u>, when a data rate change to the higher data rate has been negotiated, prior to transitioning to the higher data rate at which equalization needs to be performed. These preset values are determined by implementation specific means. After the data rate change to the higher data rate, the Upstream Port transmits <u>TS1 Ordered Sets</u> with the preset values it received. If the Downstream Port did not receive preset values in <u>Recovery.RcvrCfg</u>, after the data rate change to the higher data rate, it transmits <u>TS1 Ordered Sets</u> with the presets as follows:</p> <ul style="list-style-type: none"> For equalization at 16.0 GT/s: <u>Downstream Port 16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. For equalization at 32.0 GT/s: <u>Downstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. For equalization at 64.0 GT/s: <u>Downstream Port 64.0 GT/s Transmitter Preset field of the 64.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. <p>The preset values must be within the operable range defined in <u>§ Section 8.3.3.3</u> if reduced swing will be used by the Transmitter.</p> <p>To perform redo equalization, the Downstream Port must request speed change through <u>TS1 Ordered Sets</u> in <u>Recovery.RcvrLock</u> with the Equalization Redo bit set to 1b to inform the Upstream Port that it intends to redo equalization. An Upstream Port should advertise the higher data rate in <u>Recovery</u> if it receives <u>TS1 Ordered Sets</u> with speed change bit set to 1b, Equalization Redo bit set to 1b and it intends to operate at the higher data rate in the future.</p> |
| <p>From 2.5 GT/s or 5.0 GT/s to 32.0 GT/s Equalization</p> | <p>Equalization to 32.0 GT/s from 2.5 GT/s or 5.0 GT/s is possible only if the Link is capable of bypassing equalization to higher data rate(s) (i.e., <u>Equalization Bypass to Highest NRZ Rate Supported in 32.0 GT/s Capabilities register</u> is 1b and <u>Equalization Bypass to Highest NRZ Rate Disable in the 32.0 GT/s Control Register</u> is 0b).</p> <p>The mechanisms described here are identical for all flavors of equalization: initial or redo equalization; autonomous or software initiated.</p> <p>The Downstream Port communicates the Transmitter preset values for each Lane to the Upstream Port using 8b/10b encoding. These values are communicated using the <u>EQ TS2 Ordered Sets</u> (defined in <u>§ Section 4.2.5.1</u>) in <u>Recovery.RcvrCfg</u>, when a data rate change to the higher data rate has been negotiated, prior to transitioning to the higher data rate to perform equalization. The preset values sent in the <u>EQ TS2 Ordered Sets</u> are derived as follows:</p> <ul style="list-style-type: none"> For equalization at 32.0 GT/s: <u>Upstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry</u> corresponding to the Lane. <p>After the data rate change to the higher data rate where equalization needs to be performed, the Upstream Port transmits <u>TS1 Ordered Sets</u> with the preset values it received. The preset values must be within the operable range defined in <u>§ Section 8.3.3.3</u> if reduced swing will be used by the Transmitter.</p> <p>After the data rate change to the higher data rate where equalization needs to be performed, the Downstream Port transmits <u>TS1 Ordered Sets</u> with the preset values as follows with the assumption that the preset values must be within the operable range defined in <u>§ Section 8.3.3.3</u> if reduced swing will be used by the Transmitter:</p> |

| | |
|---|--|
| | <ul style="list-style-type: none"> If eight consecutive EQ TS2 Ordered Sets were received with supported Transmitter Preset values in the most recent transition through Recovery.RcvrCfg, the Transmitter Preset value from those EQ TS2 Ordered Sets must be used. Otherwise: Downstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry corresponding to the Lane must be used. <p>To perform redo equalization, the Downstream Port must request speed change through EQ TS1 Ordered Sets in Recovery.RcvrLock at 2.5 GT/s or 5.0 GT/s to inform the Upstream Port that it intends to redo equalization at the higher data rate. An Upstream Port should advertise the higher data rate in Recovery if it receives EQ TS1 Ordered Sets with speed change bit set to 1b and if it intends to operate at the higher data rate in the future.</p> |
| From 2.5 GT/s or 5.0 GT/s to 64.0 GT/s Equalization | Equalization to 64.0 GT/s from 2.5 GT/s or 5.0 GT/s is not supported. All equalization procedures at the 64.0 GT/s data rate, including re-equalization, must be initiated from the 32.0 GT/s data rate only. |
| Equalization at a data rate from a data rate equal to the target equalization data rate | <p>This is only possible with a redo equalization. The combinations covered here are: 8.0 GT/s equalization from 8.0 GT/s data rate, 16.0 GT/s equalization from 16.0 GT/s data rate, and 32.0 GT/s equalization from 32.0 GT/s data rate.</p> <p>In this case, the initial preset used during equalization is equal to the initial preset used during the last time the equalization was performed at the data rate where equalization is being performed.</p> <p>64.0 GT/s equalization from 64.0 GT/s is not supported.</p> |

The equalization procedure consists of up to four Phases, as described below. When operating at 8.0 GT/s or higher data rates, the Phase information is transmitted using the Equalization Control (EC) field in the TS0 or TS1 Ordered Sets.

Phase 0

This phase is executed while negotiating (and prior to) to the data rate where equalization would be performed. The preset to be used for equalization is determined as described in § Table 4-33 .

Phase 1

Both components make the Link operational enough at the current data rate to be able to exchange TS1 Ordered Sets to complete the remaining phases for the fine-tuning of their Transmitter/Receiver pairs. For the Data Rates of 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s, TS1 Ordered Sets are exchanged in this Phase. For the Data Rate of 64.0 GT/s, TS0 Ordered Sets are exchanged in this Phase and each Pseudo-Port or Port must be able to exchange the TS0 Ordered Sets for Phase 1 and Phase 2 in the Upstream Direction (from the Upstream Port to Downstream Port) reliably when it exits this Phase. It is expected that the Link will operate at a BER of less than 10^{-4} before the component is ready to move on to the next Phase. Each Transmitter uses the preset values as described in § Table 4-33 .

The Downstream Port initiates Phase 1 by transmitting TS1 Ordered Sets for Data Rates of 8.0 GT/s, 16.0 GT/s and 32.0 GT/s and TS0 Ordered Sets for the Data Rate of 64.0 GT/s with EC=01b (indicating Phase 1). The Upstream Port, after adjusting its Receiver, if necessary, to ensure that it can progress with the equalization process, receives these TS0 or TS1 Ordered Sets and transitions to Phase 1 (where it transmits TS1 Ordered Sets with EC=01b if the Data Rate is 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s or TS0 Ordered Sets with EC=01b if the Data Rate is 64.0 GT/s). The Downstream Port ensures that it can reliably receive the bit stream from the Upstream Port to continue through the rest of the Phases when it receives TS0 or TS1 Ordered Sets from the Upstream Port with EC=01b before it moves on to Phase 2.

Phase 2

In this Phase the Upstream Port adjusts the Transmitter setting of the Downstream Port along with its own Receiver setting, independently, on each Lane, to ensure it receives the bit stream compliant with the requirements in § Chapter 8. (e.g., each operational Downstream Lane has a BER less than 10^{-12} for a Data Rate of 32.0 GT/s or lower

or 10^{-6} for the Data Rate of 64.0 GT/s). If the Data Rate is 8.0 GT/s, 16.0 GT/s or 32.0 GT/s, both the Downstream Port and the Upstream Port transmit TS1 Ordered Sets. If the Data Rate is 64.0 GT/s, the Downstream Port transmits TS0 Ordered Sets until it receives TS0 Ordered Sets and then it transmits TS1 Ordered Sets whereas the Upstream Port transmits TS0 Ordered Sets. The Downstream Port initiates the move to Phase 2 by transmitting TS0 / TS1 Ordered Sets with EC=10b to the Upstream Port. The Downstream Port advertises the Transmitter coefficients and the preset it is using per the rules below in Phase 1 for preset only and in Phase 2 for preset and coefficients. The Upstream Port receives these Ordered Sets and may request different coefficient or preset settings and continue to evaluate each setting until it arrives at the best setting for operating the Downstream Lanes. After the Upstream Port has completed this Phase, it moves the Link to Phase 3 by transmitting TS1 Ordered Sets for a Data Rate of 32.0 GT/s or lower and TS0 Ordered Sets for a Data Rate of 64.0 GT/s with EC=11b to the Downstream Port.

Phase 3

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

The algorithm used by a component to adjust the transmitter of its Link partner and the evaluation of that Transmitter set-up with its Receiver set-up is implementation specific. A component may request changes to any number of Lanes and can request different settings for each Lane. Each requested setting can be a preset or a set of coefficients that meets the requirements defined in § Section 4.2.4.1 . Each component is responsible for ensuring that at the end of the fine-tuning (Phase 2 for Upstream Ports and Phase 3 for Downstream Ports), its Link partner has the Transmitter setting in each Lane that will cause the Link to meet the requirements in § Chapter 8..

A Link partner receiving the request to adjust its Transmitter must evaluate the request and act on it. If a valid preset value is requested and the Transmitter is operating in full swing mode, it must be reflected in the Transmitter set-up and subsequently in the preset and coefficient fields of the TS1 Ordered Set that the Link partner transmits. If a preset value is requested, the Transmitter is operating in reduced swing mode, and the requested preset is supported as defined in § Section 8.3.3.3 it must be reflected in the Transmitter set-up and subsequently in the preset and coefficient fields of the TS1 Ordered Set that the Link partner transmits. Transmitters operating in reduced swing mode are permitted to reject preset requests that are not supported as defined in § Section 8.3.3.3 . A request for adjusting the coefficients may be accepted or rejected. If the set of coefficients requested for a Lane is accepted, it must be reflected in the Transmitter set-up and subsequently in the transmitted TS1 Ordered Sets. If the set of coefficients requested for a Lane is rejected, the Transmitter set-up is not changed, but the transmitted TS1 Ordered Sets must reflect the requested coefficients along with the Reject Coefficient Values bit set to 1b. In either case of responding to a coefficient request, the preset field of the transmitted TS1 Ordered Sets is not changed from the last preset value that was transmitted. A request for adjusting the coefficients may be rejected by the Link partner only if the set of coefficients requested is not compliant with the rules defined in § Section 4.2.4.1 .

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

If a Lane receives a Transmitter Preset value (from a TS0 , TS1 or TS2 sequence) with a Reserved or unsupported value in Polling.Compliance , Loopback , or Phase 0 or Phase 1 of Recovery.Equalization , then the Lane is permitted to use any supported Transmitter preset setting in an implementation specific manner. The Reserved or unsupported Transmitter preset value is transmitted in any subsequent compliance patterns or Ordered Sets, and not the implementation specific Transmitter preset value chosen by the Lane. For example, if a Lane of an Upstream Port receives a Transmitter preset value 1111b (Reserved) with the EQ TS2 Ordered Sets it receives in Recovery.RcvrCfg , it is permitted to use any supported Transmitter preset value for its transmitter setting after changing the data rate to 8.0 GT/s, but it must transmit 1111b as its Transmitter preset value in the TS1 Ordered Sets it transmits in Phase 0 and Phase 1 of Recovery.Equalization .

In the Loopback state, the Loopback Lead is responsible for communicating the Transmitter and Receiver settings it wants the Follower to use through the EQ TS1 Ordered Sets it transmits in the 2.5 GT/s or 5.0 GT/s data rate, and the

preset or coefficient settings it wants the device under test to operate under in the TS1 Ordered Sets it transmits in the 8.0 GT/s or higher data rate. Similarly, if the Polling.Compliance state for 8.0 GT/s or higher Data Rates is entered through TS1 Ordered Sets, the entity that is performing the test is required to send the appropriate settings in EQ TS1 Ordered Sets and presets for the device under test to operate with, according to the mechanism defined in § Section 4.2.7.2.

IMPLEMENTATION NOTE: EQUALIZATION EXAMPLE §

Some examples are presented in this note to help explain Link equalization. This is not a complete listing of all allowable equalization scenarios.

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

§ Figure 4-62 shows an example flow for 64.0 GT/s after the Link completes the 32.0 GT/s equalization with the use of TS0 and TS1 Ordered Sets.

Note that some transitions may not be covered; such as the transition from receiving TS0 to receiving TS1 in a given sub-state.

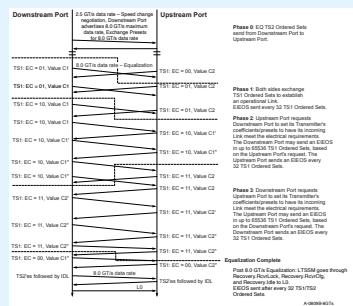


Figure 4-60 8.0 GT/s Equalization Flow §

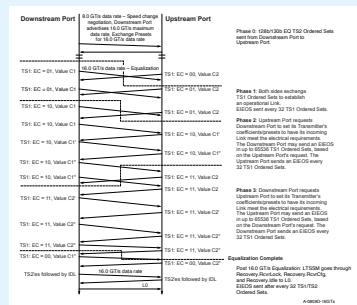


Figure 4-61 16.0 GT/s Equalization Flow §

Base 6.4 vs Base 6.3

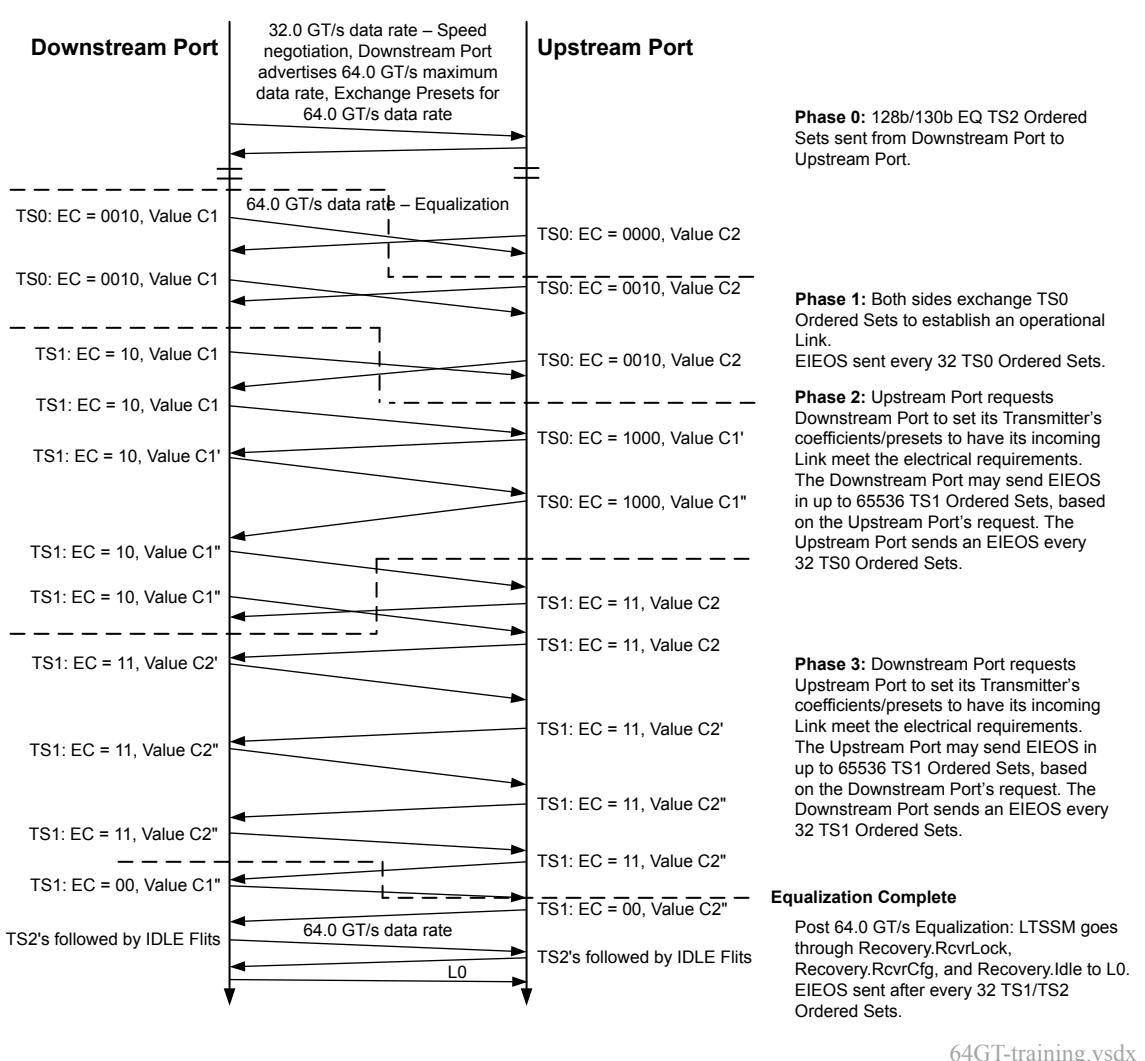


Figure 4-62 64.0 GT/s Equalization Flow §

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: EQUALIZATION BYPASS EXAMPLE §

The following flow-chart provides an example flow where a Link may bypass equalization at lower Data Rates and go to the highest supported NRZ rate for equalization. For example, when n=5, the Link can train to L0 in Gen 1 data rate, establish that all components (including Retimers, if any) can bypass equalization to Gen 5, change the data rate to Gen 5 and just perform equalization at Gen 5.

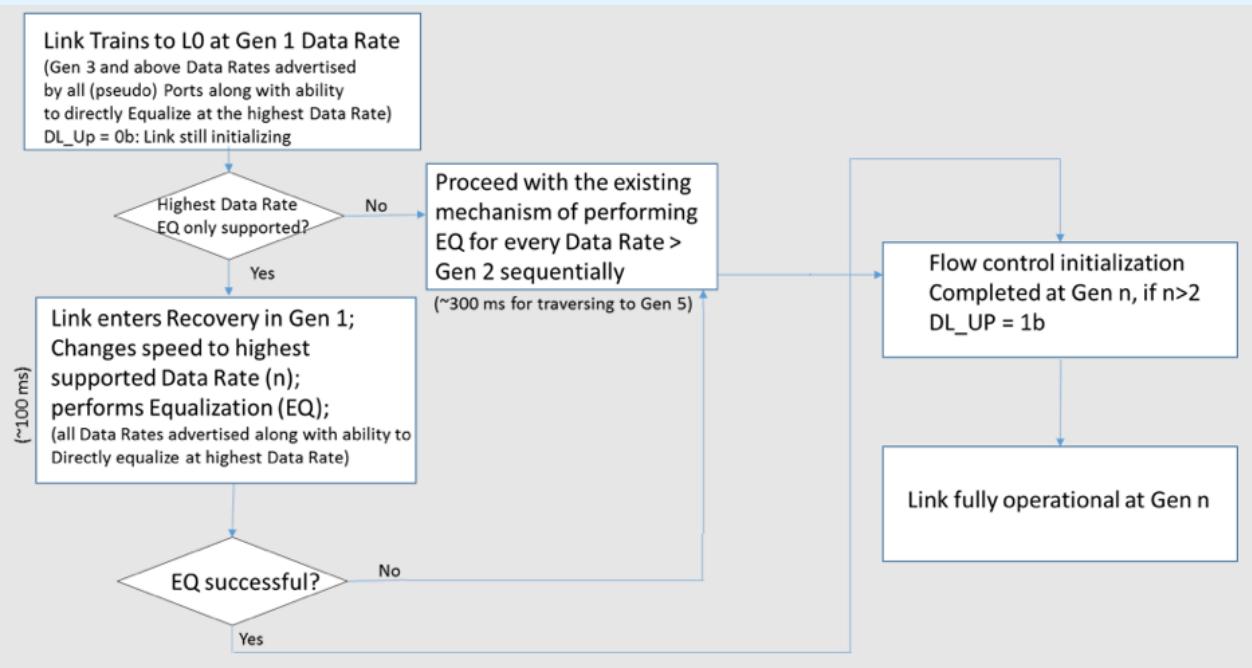


Figure 4-63 Equalization Bypass Example §

4.2.4.1 Rules for Transmitter Coefficients §

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

- C₋₂, C₋₁, and C₊₁** are the coefficients used in the FIR equation and represent the 2nd pre-cursor, pre-cursor and post-cursor, respectively. The pre-cursors and post-cursor values communicated in the TS1 Ordered Sets represent their absolute values. **C₀** represents the cursor coefficient setting and is a positive entity. The coefficient **C₋₂ is used only for 64.0 GT/s and higher Data Rates.**
- The sum of the absolute values of the coefficients defines the FS (Full Swing; $FS = |C_{-2}| + |C_{-1}| + C_0 + |C_{+1}|$). FS is advertised to the Link partner in Phase 1. The Transmitter FS range is defined below:
 - FS ∈ {24, ..., 63} (i.e., FS must have a value from 24 through 63) for full swing mode.
 - FS ∈ {12, ..., 63} for reduced swing mode.
 - C₋₂ is set to 0 for Data Rates lower than 64.0 GT/s.

3. A Transmitter advertises its LF (Low Frequency) value during Phase 1.
 - This corresponds to the minimum differential voltage that can be generated by the Transmitter which is LF/FS times the Transmitters maximum differential voltage. The Transmitter must ensure that LF meets the electrical requirements defined in § Section 8.3.3.9 for $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$.
4. The following rules must be satisfied before a set of coefficients can be requested of the Link partner's Transmitter. Upon reception of an update request for TX coefficient settings, a Port must verify that the new request meets the following conditions and reject the request if any of following conditions are violated:
 - a. $|C_{-1}| \leq \text{Floor}(FS/4)$
 - b. $|C_{-1}| + |C_{-2}| + C_0 + |C_{+1}| = FS$
(Do not allow peak power to change with adaptation)
 - c. $C_0 - |C_{-1}| + |C_{-2}| - |C_{+1}| \geq LF$
 - d. $|C_{-2}| \leq \text{Floor}(FS/8)$

4.2.4.2 Encoding of Presets §

Definition of the Transmitter and Receiver Preset Hints appears in § Chapter 8. . The encoding for the Transmitter Preset and Receiver Preset Hint are provided in § Table 4-34 and § Table 4-35 . There are two classes of presets: P0 through P10 are defined for data rates of 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s whereas Q0 through Q10 are defined for the data rate of 64.0 GT/s. Even though 8 of the 11 presets across these two sets overlap, it is possible that the same preset values may get different encodings between the two classes. Receiver Preset Hints are optional and only defined for the 8.0 GT/s data rate.

Table 4-34 Transmitter Preset Encoding §

| Encoding | Preset Number in § Table 8-1 | Preset Number in § Table 8-2 |
|---------------------|------------------------------|------------------------------|
| 0000b | P0 | Q0 |
| 0001b | P1 | Q1 |
| 0010b | P2 | Q2 |
| 0011b | P3 | Q3 |
| 0100b | P4 | Q4 |
| 0101b | P5 | Q5 |
| 0110b | P6 | Q6 |
| 0111b | P7 | Q7 |
| 1000b | P8 | Q8 |
| 1001b | P9 | Q9 |
| 1010b | P10 | Q10 |
| 1011b through 1111b | Reserved | Reserved |

*Table 4-35 Receiver Preset Hint
Encoding for 8.0 GT/s §*

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

IMPLEMENTATION NOTE: QUANTIZATION ERRORS AT 64.0 GT/S §

Due to the tighter preset tolerance at 64.0 GT/s some FS values will result in a quantization error larger than is allowed for the specified tolerance of the presets in § Table 8-2 of § Section 8.3.3.3 . The implementation must compensate for any quantization error from its selection of FS to meet the specified preset accuracy.

4.2.5 Link Initialization and Training §

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

- configuring and initializing the Link
- supporting normal packet transfers
- supported state transitions when recovering from Link errors
- restarting a Port from low power states.

The following are discovered and determined during the training process:

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

Training does:

- Link data rate negotiation
- Bit lock per Lane

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

4.2.5.1 Training Sequences §

Training sequences are composed of Ordered Sets used for initializing bit alignment, Symbol alignment and to exchange Physical Layer parameters.

1. When the data rate is 2.5 GT/s or 5.0 GT/s, Ordered Sets are never scrambled but are always 8b/10b encoded.
2. When the data rate is between 8.0 GT/s and 32.0 GT/s, the 128b/130b encoding is used and Symbols may or may not be scrambled, according to the rules in § Section 4.2.2.4 .
3. When the data rate is 64.0 GT/s the 1b/1b encoding is used and Symbols may or may not be scrambled, according to the rules in § Section 4.2.3.1.2 .

Training sequences (TS0 , TS1 , TS2 , Modified TS1 , or Modified TS2) are transmitted consecutively and can only be interrupted by SKP Ordered Sets (see § Section 4.2.8) or, for data rates other than 2.5 GT/s, EIEOS (see § Section 4.2.5.3).

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

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

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

Components that intend to either negotiate alternate protocols or pass a Training Set Message must use Modified TS1/ TS2 Ordered Sets instead of standard TS1 / TS2 Ordered Sets in Configuration.Lanenum.Wait , Configuration.Lanenum.Accept , and Configuration.Complete substates. In order to be eligible to send the Modified TS1/ TS2 Ordered Sets , components must set the Enhanced Link behavior Control bits (bit 7:6 of Symbol 5) in TS1 and TS2 Ordered Sets to 11b in Polling.Active , Polling.Configuration , Configuration.Linkwidth.Start , and Configuration.Linkwidth.Accept substates and follow through the steps outlined on transition to Configuration.Lanenum.Wait substate when LinkUp =0b. If the Link partner does not support Modified TS1/TS2 Ordered Sets , then starting with Configuration.Lanenum.Wait , the standard TSs should stop sending 11b in the Enhanced Link Behavior Control field and switch to the appropriate encodings.

When using 8b/10b encoding, modified TS1 or modified TS2 Ordered Sets are considered consecutive only if all Symbols matches the corresponding Symbols of the previous modified TS1 or modified TS2 Ordered Sets and the parity in Symbol 15 matches with the expected value. Symbols 8-14 must be identical in each Modified TS1/TS2 Ordered Sets across all Lanes of a Link.

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

With 1b/1b encoding, TS0, TS1 and TS2 Ordered Sets consist of the first 8 symbols of the Ordered Set being replicated in the second 8 symbols of the Ordered Set, as shown in § Table 4-39 and § Table 4-40. The TS0 Ordered Set is used with 1b/1b encoding during the Recovery.Equalization, as described in the LTSSM section. Each UI in the TS0 Ordered Set is either voltage level 0 or 3 with PAM-4 signaling, as shown in § Figure 4-24. Hence, the scrambled symbols in TS0 Ordered Sets are **Half scrambled**; the odd bit positions are scrambled whereas even bit position j is identical to the odd bit position $j+1$. Precoding is bypassed in both the Transmitter and Receiver.

In 1b/1b encoding, **valid 1b/1b TS0, TS1, or TS2 Ordered Set** is defined as a received TS0, TS1, or TS2 Ordered Set where either half is valid:

- The first half is valid if Symbol 0 is a valid TS0 / TS1 / TS2 encoding, Symbols 0 to 7 passes its parity check after decoding Gray code and descrambling, and Symbol 7 is not equal to a DC balance pattern prior to performing gray code and descrambling.
- The second half is valid if Symbol 8 is a valid TS0 / TS1 / TS2 encoding, Symbols 8 to 15 passes its parity check after decoding Gray code and descrambling, and symbol 15 is not equal to a DC balance pattern prior to performing gray code and descrambling.
- Both halves are valid if Symbols 0 to 6 are identical to Symbols 8 to 14 after decoding Gray code and descrambling, and Symbols 7 and 15 are DC balance symbols prior to decoding Gray code and descrambling.
- The type of the Ordered Set (TS0, TS1, or TS2) is determined by the first symbol of the valid half.
- The even bits in the half-scrambled Symbols must be ignored after descrambling in the Receiver. (Note: because the gray code was done by forcing the even bit to be identical to the odd bit in the Transmitter, the even bit position may not be identical to the odd bit position on the Receiver after descrambling.)

IMPLEMENTATION NOTE: HALF SCRAMBLING EXAMPLE §

Since the half-scrambled symbols are not fully gray-coded on the transmitter, rather the even bit is made identical to odd bit to have that UI be either a voltage level 0 or voltage level 3, it is possible that an even bit position in the half-scrambled symbol in the Receiver after gray-coding and descrambling may not match what was transmitted. For example, the Transmitter intends to send 80h prior to scrambling. After scrambling, the symbol becomes 57h and the gray-coded value by forcing the even bit to be identical to the odd bit becomes 03h. On the Receiver side, 03h after gray coding remains at 03h and after descrambling becomes 94h. Here bit 0 is different between the transmitted side 80h vs. the receive side 94h. However, all the odd bit positions that carry meaningful information are identical. By AND'ing 94h with AAh, we get 80h which was sent.

With 1b/1b encoding, two TS0, TS1 or TS2 Ordered Sets are considered consecutive if all of the following conditions apply to both the Ordered Sets:

- both Ordered Sets are of the same type
- they arrived one after the other, even though they may be separated by one or more SKP Ordered Set(s)
- each ordered set has a valid half and the first 7 symbols of the valid halves of each of the Ordered Set are identical, with Reserved bits treated as described below.

An invalid Ordered Set is a Training Set that does not pass the checks in § Section 4.2.5.1.

Reserved bits in TS0, TS1, TS2, Modified TS1 or Modified TS2 Ordered Sets must be handled as follows:

- The Transmitter must transmit 0s for Reserved bits.
- The Receiver:

- must not determine that a TS0, TS1, TS2, Modified TS1 or Modified TS2 Ordered Set is invalid based on the received value of Reserved bits
- must use the received value of Reserved bits for the purpose of a parity computation if the Reserved bits are included in a parity calculation
- may optionally compare the received value of Reserved bits within Symbols that are explicitly called out as being required to be identical in TS0, TS1, TS2, Modified TS1, or Modified TS2 Ordered Sets to determine if they are consecutive
- must not otherwise take any functional action based on the value of any received Reserved bits

When using 128b/130b or 1b/1b encoding, Transmitters are required to track the running DC Balance of the bits transmitted on the wire (after Gray code in 1b/1b, scrambling, and if turned on: precoding) that constitute the TS0, TS1, and TS2 Ordered Sets only. The running DC Balance is the difference between the number of 1s transmitted and the number of 0s transmitted in 8.0 GT/s, 16.0 GT/s or 32.0 GT/s Data Rates. The running DC Balance is the running sum of the DC balance values assigned to each voltage level in each UI, as shown in § Figure 4-24, for 64.0 GT/s Data Rate. Each Lane must track its running DC Balance independently and be capable of tracking a difference of at least +/-511 bits. Any counters used must saturate at their limit (not roll-over) and continue to track reductions after their limit is reached. For example, a counter that can track a difference of 511 will saturate at 511 if a difference of 513 is detected, and then change to 509 if the difference is reduced by 2 in the future.

The running DC Balance is set to 0 by two events:

1. The Transmitter exiting Electrical Idle;
2. Transmission of an EIEOS following a Data Block.

For every TS1 or TS2 Ordered Set transmitted with 128b/130b encoding, Transmitters must evaluate the running DC Balance and transmit one of the DC Balance Symbols defined for Symbols 14 and 15 as defined by the algorithm below. If the number of 1s needs to be reduced, the DC Balance Symbols 20h (for Symbol 14) and 08h (for Symbol 15) are transmitted. If the number of 0s needs to be reduced, the DC Balance Symbols DFh (for Symbol 14) and F7h (for Symbol 15) are transmitted. If no change is required, the appropriate TS1 or TS2 Identifier Symbol is transmitted. Any DC Balance Symbols transmitted for Symbols 14 or 15 bypass scrambling, while TS1 and TS2 Identifier Symbols follow the standard scrambling rules. The following algorithm must be used to control the DC Balance with 128b/130b encoding:

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

Receivers are permitted, but not required, to check Symbols 14 and 15 for the following values when determining whether a TS1 or TS2 Ordered Set is valid with 128b/130b encoding:

- The appropriate TS Identifier Symbol after de-scrambling, or
- a valid DC Balance Symbol of DFh or 20h before de-scrambling for Symbol 14, or
- a valid DC Balance Symbol of F7h or 08h before de-scrambling for Symbol 15.

If a Receiver receives a DC balance pattern in Symbol 14 in 128b/130b encoding, it is possible that the pattern is scrambled (and precoded). Thus, if the Receiver is performing this optional check, it must keep descrambler and receive precoding running for checking Symbol 15, which can be either scrambled (and precoded) or the DC balance pattern.

IMPLEMENTATION NOTE: SYNC HEADER AND DC BALANCE §

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

For every TS0 , TS1 or TS2 Ordered Set transmitted with 1b/1b encoding, Transmitters must evaluate the running DC Balance and transmit one of the DC Balance Symbols defined for Symbols 7 and 15 as defined by the algorithm below. Any DC Balance Symbols transmitted in Symbols 7 and 15 bypass scrambling, while the byte-wise parity, if sent in Symbols 7 and 15, follows the standard scrambling rules.

- If the running DC Balance was $>+47$ at the start of the TS Ordered Set, transmit 00h in Symbols 7 and 15 without scrambling.
- If the running DC Balance was <-47 at the start of the TS Ordered Set, transmit FFh in Symbols 7 and 15 without scrambling.
- Else, transmit the even byte parity of Symbols 0-6 (pre-scrambling) in Symbols 7 and 15 with scrambling.

While using TS0 Ordered Sets, the coefficients are not the full 6-bit value but adjusted to the appropriate number of bits that the corresponding coefficients, based on the constraints. For example, 3 bits are enough for the 2nd pre-cursor since it cannot be higher than 7.

The definitions of Hot_Reset_Request , Disable_Link_Request , Loopback_Request , and Compliance_Receive_Request depend on the encoding being used.

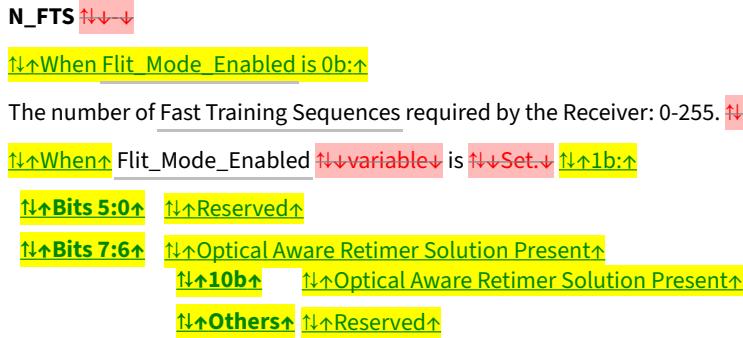
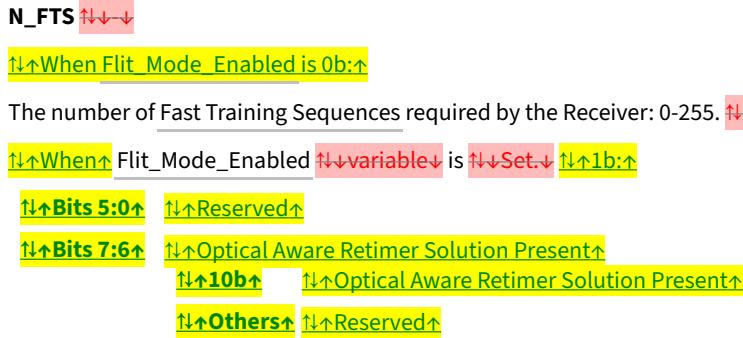
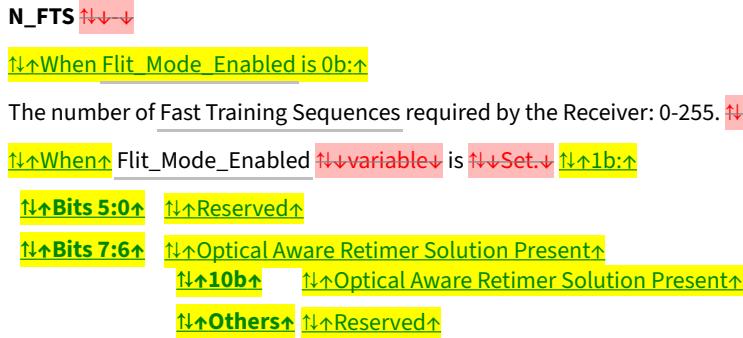
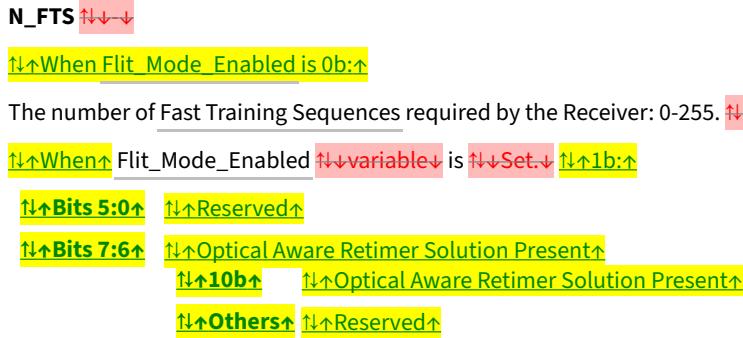
- When operating with 8b/10b or 128b/130b encoding, **Hot_Reset_Request** , **Disable_Link_Request** , **Loopback_Request** , or **Compliance_Receive_Request** are generated by asserting the appropriate Training Control bit. Hot_Reset_Request bit , Disable_Link_Request bit , and Loopback_Request bit asserted are mutually exclusive. Only one of those bits is permitted to be asserted at a time when transmitted on a configured Link or on all Lanes during Configuration. If more than one of Hot_Reset_Request bit , Disable_Link_Request bit , or Loopback_Request bit are asserted at the same time, the Link behavior is undefined. In some cases, Compliance_Receive_Request bit is permitted to be asserted when Loopback_Request bit is also asserted. The Link behavior is undefined when Compliance_Receive_Request bit is asserted and either Hot_Reset_Request bit or Disable_Link_Request bit is also asserted.
- When operating with 1b/1b encoding, Hot_Reset_Request , Disable_Link_Request , Loopback_Request , and Compliance_Receive_Request are generated using the appropriate Training Control field encoding. See § Section 4.2.5.1 , § Table 4-39 for the Training Control field encodings.

The TS1 Ordered Set's Retimer Equalization Extend bit is always set to 0b when transmitted by an Upstream Port or Downstream Port. Retimers set the bit to 1b as described in § Section 4.3.8.2 .

Table 4-36 TS1 Ordered Set in 8b/10b and 128b/130b Encoding §

| TS1 Ordered Set in 8b/10b and 128b/130b Encoding | |
|--|--|
| Symbol Number | Description |
| TS1 Identifier | |
| 0 | <ul style="list-style-type: none"> • When operating at 2.5 or 5.0 GT/s: COM (K28.5) for Symbol alignment. |

TS1 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description |
|---------------|---|
| | <ul style="list-style-type: none"> When operating at 8.0 GT/s or above: Encoded as 1Eh (TS1 Ordered Set). |
| 1 | <p>Link Number</p> <ul style="list-style-type: none"> Ports that do not support 8.0 GT/s or above: 0-255, PAD. Downstream Ports that support 8.0 GT/s or above: 0-31, PAD. Upstream Ports that support 8.0 GT/s or above: 0-255, PAD. When operating at 2.5 or 5.0 GT/s: PAD is encoded as K23.7. When operating at 8.0 GT/s or above: PAD is encoded as F7h. |
| 2 | <p>Lane Number within Link</p> <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: 0-31, PAD. PAD is encoded as K23.7. When operating at 8.0 GT/s or above: 0-31, PAD. PAD is encoded as F7h. |
| 3 | <p>N_FTS </p> <p> When Flit_Mode_Enabled is 0b: When Flit_Mode_Enabled is variable is Set, 1b: Bits 5:0, Reserved Bits 7:6, Optical Aware Retimer Solution Present 10b, Optical Aware Retimer Solution Present Others, Reserved</p> <p>The number of Fast Training Sequences required by the Receiver: 0-255.  Reserved when the</p> <p>ECN: Base 6.3 Optical </p> |
| 4 | <p>Data Rate Identifier</p> <p>Bit 0 Flit Mode Supported bit :</p> <ul style="list-style-type: none"> 0b Flit Mode is not Supported (i.e., In the transmitter: either Flit Mode Supported is Clear (see PCI Express Capabilities Register) or Flit Mode Disable is Set (see Flit Mode Supported bit) 1b Flit Mode is supported (i.e., both Flit Mode Supported is Set (see PCI Express Capabilities Register) and Flit Mode Disable is Clear (see Flit Mode Supported bit) <p>Bits 5:1 Supported Link Speeds :</p> <ul style="list-style-type: none"> Non-Flit Mode Encodings (valid during Flit Mode negotiation or when Flit Mode is not negotiated) 0 0001b Only 2.5 GT/s supported 0 0011b Only 2.5 and 5.0 GT/s supported 0 0111b Only 2.5, 5.0, and 8.0 GT/s supported 0 1111b Only 2.5, 5.0, 8.0, and 16.0 GT/s supported 1 1111b Only 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s supported Others Reserved in Non-Flit Mode |

TS1 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description |
|----------------|---|
| | Additional encodings permitted after Flit Mode is negotiated by all Link Partners after the first entry to Configuration.Complete from Detect or in the following states when the device supports Flit Mode: Polling.Active as a Receiver (see § Section 4.2.7.2.1), Configuration.Linkwidth.Start (see § Section 4.2.7.3.1), Recovery.Equalization (see § Section 4.2.7.4.1 and § Section 4.2.7.4.2) and Loopback (see § Section 4.2.7.10): |
| 1 0111b | 2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s supported if component had transmitted 1 1111b in Supported Link Speeds in Polling and Configuration states since entering the Polling state. |
| Others | Reserved |
| Bit 6 | <i>Autonomous Change / Selectable De-emphasis :</i> <ul style="list-style-type: none"> Downstream Ports: This bit is defined for use in the following LTSSM states: Polling.Active, Configuration.Linkwidth.Start, and Loopback.Entry. In all other LTSSM states, it is Reserved. Upstream Ports: This bit is defined for use in the following LTSSM states: Polling.Active, Configuration, Recovery, and Loopback.Entry. In all other LTSSM states, it is Reserved. |
| Bit 7 | <i>speed_change / SRIS Clocking</i> <ul style="list-style-type: none"> Downstream Ports: <ul style="list-style-type: none"> In Configuration and Loopback.Entry States: <ul style="list-style-type: none"> When LinkUp=0b: A 1b indicates that the Link will operate in SRIS clocking; a 0b indicates either common clocking or SRNS clocking. The Downstream Port uses this bit to communicate the type of clocking to the Retimer(s), if any, as well as the Upstream Port so that the correct (Control) SKP Ordered Set frequency can be selected Else: this bit is Reserved In Recovery.RcvrLock : speed_change In other states: Reserved Upstream Ports: speed_change . This bit can be set to 1b only in the Recovery.RcvrLock LTSSM state. In all other LTSSM states, it is Reserved |
| 5 | Training Control |
| | Bit 0 <i>Hot_Reset_Request bit</i> <ul style="list-style-type: none"> 0b Deassert 1b Assert |
| | Bit 1 <i>Disable_Link_Request bit</i> <ul style="list-style-type: none"> 0b Deassert 1b Assert |
| | Bit 2 <i>Loopback_Request bit</i> <ul style="list-style-type: none"> 0b Deassert 1b Assert |
| | Bit 3 <i>Disable Scrambling bit</i> (2.5 GT/s and 5.0 GT/s data rates) <ul style="list-style-type: none"> 0b Deassert 1b Assert Reserved (other data rates) |
| 5 | Bit 4 <i>Compliance_Receive_Request bit</i> (5.0 GT/s and above data rates, optional at 2.5 GT/s) |

Base 6.4 vs Base 6.3

TS1 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description | | |
|---|---|---|---|
| | 0b | Deassert | |
| | 1b | Assert | |
| | | Ports that support 5.0 GT/s and higher data rate(s) must implement <u>Compliance_Receive_Request</u> . Ports that support only 2.5 GT/s data rate may optionally implement <u>Compliance_Receive_Request</u> . If not implemented, this bit is Reserved. | |
| | Bit 5 | Transmit Modified Compliance Pattern in Loopback . This bit is defined for use in Loopback by the Loopback Lead when 32.0 GT/s or higher data rates are supported. See § <u>Section 4.2.7.10.1</u> . In all other cases, this bit is Reserved. | |
| | Bit 7:6 | Enhanced Link Behavior Control | |
| | 00b | Full Equalization Required Modified TS1/TS2 Ordered Sets not supported | |
| | 01b | Equalization Bypass to Highest NRZ Rate Support Modified TS1/TS2 Ordered Sets not supported. Indicates intention to perform 32.0 GT/s equalization when set by <u>Loopback Lead</u> . See § <u>Section 4.2.4</u> and § <u>Section 4.2.7.10.1</u> . | |
| | 10b | No Equalization Needed Modified TS1/TS2 Ordered Sets not supported A device advertising this capability must support Equalization Bypass to Highest NRZ Rate Support. See § <u>Section 4.2.4</u> . | |
| | 11b | Modified TS1/TS2 Ordered Sets supported Equalization bypass options specified in Modified TS1/TS2 Ordered Sets. | |
| | | These bits are defined for use in Polling and Configuration when <u>LinkUp =0b</u> and 32.0 GT/s or higher data rates are supported and in <u>Loopback by the Loopback Lead</u> when 32.0 GT/s or higher data rates are supported. In all other cases, these bits are Reserved. | |
| 6 | When operating at 2.5 or 5.0 GT/s: | | |
| | <ul style="list-style-type: none"> Standard <u>TS1</u> Ordered Sets encode this Symbol as a <u>TS1 Identifier</u>, D10.2 (4Ah). Compliance TS1 Ordered Sets encode the symbol as follows: <table> <tr> <td>Bits 7:0</td> <td>Compliance Setting Number defined in § <u>Table 4-61</u>. See § <u>Section 4.2.7.2.2</u> for when to send Compliance TS1 Ordered Sets .</td> </tr> </table> | | Bits 7:0 |
| Bits 7:0 | Compliance Setting Number defined in § <u>Table 4-61</u> . See § <u>Section 4.2.7.2.2</u> for when to send Compliance TS1 Ordered Sets . | | |
| <ul style="list-style-type: none"> <u>EQ</u> TS1 Ordered Sets encode this Symbol as follows: <ul style="list-style-type: none"> For Equalization at 8.0 GT/s Data Rate: <table> <tr> <td>Bits 2:0</td> <td>Receiver Preset Hint. See § <u>Section 4.2.4.2</u> .</td> </tr> </table> | | Bits 2:0 | Receiver Preset Hint . See § <u>Section 4.2.4.2</u> . |
| Bits 2:0 | Receiver Preset Hint . See § <u>Section 4.2.4.2</u> . | | |
| <table> <tr> <td>Bit 6:3</td> <td>Transmitter Preset . See § <u>Section 4.2.4.2</u> .</td> </tr> </table> | | Bit 6:3 | Transmitter Preset . See § <u>Section 4.2.4.2</u> . |
| Bit 6:3 | Transmitter Preset . See § <u>Section 4.2.4.2</u> . | | |
| <table> <tr> <td>Bit 7</td> <td>Set to 1b.</td> </tr> </table> | | Bit 7 | Set to 1b. |
| Bit 7 | Set to 1b. | | |
| <ul style="list-style-type: none"> For Equalization at 32.0 GT/s or higher Data Rate: <table> <tr> <td>Bit 0</td> <td>Transmitter Precode Request - See § <u>Section 4.2.2.5</u> . This bit has no defined usage in receivers at this time.</td> </tr> </table> | | Bit 0 | Transmitter Precode Request - See § <u>Section 4.2.2.5</u> . This bit has no defined usage in receivers at this time. |
| Bit 0 | Transmitter Precode Request - See § <u>Section 4.2.2.5</u> . This bit has no defined usage in receivers at this time. | | |
| <table> <tr> <td>Bit 2:1</td> <td>Reserved</td> </tr> </table> | | Bit 2:1 | Reserved |
| Bit 2:1 | Reserved | | |
| <table> <tr> <td>Bit 6:3</td> <td>Transmitter Preset . See § <u>Section 4.2.4.2</u> .</td> </tr> </table> | | Bit 6:3 | Transmitter Preset . See § <u>Section 4.2.4.2</u> . |
| Bit 6:3 | Transmitter Preset . See § <u>Section 4.2.4.2</u> . | | |
| <table> <tr> <td>Bit 7</td> <td>Set to 1b.</td> </tr> </table> | | Bit 7 | Set to 1b. |
| Bit 7 | Set to 1b. | | |
| When operating at 8.0 GT/s or higher data rate: | | | |

Base 6.4 vs Base 6.3

TS1 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description | |
|---------------|---|--|
| | Bit 1:0 Equalization Control (EC). These bits are only used in the Recovery.Equalization and Loopback LTSSM states. See § Section 4.2.7.4.2 and § Section 4.2.7.10 . In all other LTSSM states, they must be set to 00b. | |
| | Bit 2 Reset EIEOS Interval Count . This bit is defined for use in the Recovery.Equalization LTSSM state. See § Section 4.2.7.4.2 and § Section 4.2.5.3 . In all other LTSSM states, it is Reserved. | |
| | Bit 6:3 Transmitter Preset . See § Section 4.2.4 and § Section 4.2.7 . | |
| | Bit 7 Use Preset / Equalization Redo . This bit is defined for use in the Recovery.Equalization , Recovery.RcvrLock and Loopback LTSSM states. See § Section 4.2.7.4.1 , § Section 4.2.7.4.2 and § Section 4.2.7.10 . In all other LTSSM states, it is Reserved. | |
| 7 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: <u>TS1 Identifier</u>. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or higher: <ul style="list-style-type: none"> Bit 5:0 FS when the EC field of Symbol 6 is 01b (see § Section 4.2.4.1). Otherwise, Pre-cursor Coefficient for the current data rate of operation. Bit 6 Transmitter Precoding On . This bit is defined for use in the Recovery state for use at 32.0 GT/s or higher. See § Section 4.2.2.5 . In all the other cases, it is Reserved. Bit 7 Retimer Equalization Extend bit. This bit is defined for use in the Recovery.Equalization LTSSM state when operating at 16.0 GT/s or higher data rate. In all other LTSSM states and when operating at 8.0 GT/s, it is Reserved. | |
| 8 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: <u>TS1 Identifier</u>. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or higher data rate: <ul style="list-style-type: none"> Bit 5:0 LF when the EC field of Symbol 6 is 01b (see § Section 4.2.4.1). Otherwise, Cursor Coefficient for the current data rate of operation. Bit 7:6 Reserved. | |
| 9 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: <u>TS1 Identifier</u>. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or higher data rate: <ul style="list-style-type: none"> Bit 5:0 Post-cursor Coefficient for the current data rate of operation. Bit 6 Reject Coefficient Values bit . This bit can only be set to 1b in specific Phases of the Recovery.Equalization LTSSM State. See § Section 4.2.7.4.2 . In all other LTSSM states, it must be set to 0b. Bit 7 Parity (P). This bit is the even parity computed over all bits of Symbols 6, 7, and 8 and bits 6:0 of Symbol 9 (i.e., XOR of all covered bits). Receivers must calculate the parity of the received bits and compare it to the received Parity bit. Received <u>TS1 Ordered Sets</u> are valid only if the calculated and received Parity match. | |
| 10 - 13 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: <u>TS1 Identifier</u>. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or above: <u>TS1 Identifier</u>. Encoded as 4Ah. | |
| 14 - 15 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: <u>TS1 Identifier</u>. Encoded as D10.2 (4Ah). | |

TS1 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description |
|---------------|--|
| | <ul style="list-style-type: none">When operating at 8.0 GT/s or above: TS1 Identifier (encoded as 4Ah) or a DC Balance Symbol. |

IMPLEMENTATION NOTE: EXPECTED USAGE OF THE 64.0 GT/S SUPPORTED LINK SPEEDS ENCODING §

The TS1 and TS2 Ordered Sets for 8b/10b and 128b/130b encoding are permitted to have the Supported Link Speeds field encoded as 1 0111b (indicating 2.5 GT/s to 64.0 GT/s support, inclusive) in the following scenarios:

1. Initial Link Training to L0 (Configuration.Complete and beyond)

The Link is training to L0 and Flit Mode has been negotiated during the training procedure (i.e., when LinkUp =0b). In this scenario the 1 0111b encoding is permitted to be used once the Link enters Configuration.Complete.

2. Configuration.Linkwidth.Start (as a Receiver only)

When the Link transitions from Configuration.Linkwidth.Start to Loopback when LinkUp =0b, the Loopback Lead is permitted to transmit TS1 Ordered Sets with the Supported Link Speeds advertising 1 0111b if it has prior knowledge that the Loopback Follower (i.e., the Receiver, which is still in Configuration.Linkwidth.Start) supports 64.0 GT/s. How the Loopback Lead obtains the prior knowledge of 64.0 GT/s support is outside the scope of the specification.

3. Recovery.Equalization when LinkUp =0b

Advertising the 1 0111b encoding in Phase 1 of Recovery.Equalization when “Equalization for Loopback” is being performed. More specifically, this is the test scenario where Recovery.Equalization is entered from Loopback, the Link is equalized at the current data rate and then the Link returns to Loopback. In the case where the data rate is less than required (i.e., Link is equalized for 32.0 GT/s but 64.0 GT/s is needed), the 1 0111b setting will be used to indicate to the Link partner the ability to transition to 64.0 GT/s upon re-entry to Loopback (after which an equalization procedure at 64.0 GT/s will be performed followed by a return to Loopback).

4. Loopback

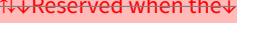
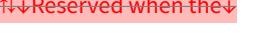
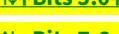
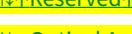
When Loopback is entered through Configuration.Linkwidth.Start and LinkUp =0b, the Loopback Lead is permitted to use the 1 0111b encoding in the TS1 Ordered sets that it transmits upon entry to Loopback.Entry (prior to the EIOSQ that proceeds Electrical Idle and the speed change).

5. Polling.Active (as a Receiver only)

Test apparatus that understands the capabilities of the Port (i.e., that the Port being exercised is capable of 64.0 GT/s operation) may transmit TS1 Ordered Sets with the Supported Link Speeds field set to the 64.0 GT/s supported encoding (1 0111b), and the Flit Mode Supported bit set (1b) in Polling.Active when the TS1 Ordered Sets also have Compliance_Receive_Request asserted and Loopback_Request deasserted. A Port that transitions to Polling.Compliance from Polling.Active because it received the appropriate TS1 Ordered Sets in Polling.Active will accept 1 0111b as a valid Supported Link Speeds encoding (and consider it in its determination of the highest common data rate) if it supports the 64.0 GT/s data rate.

Table 4-37 TS2 Ordered Set in 8b/10b and 128b/130b Encoding §

TS2 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description | | | | | | | | |
|----------------|--|----------------|-------------------------|----------------|---------------------------------|----------------|---------------------------------------|----------------|---|
| 0 | <p>TS2 Identifier</p> <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: COM (<u>K28.5</u>) for Symbol alignment. When operating at 8.0 GT/s or above: Encoded as 2Dh (TS2 Ordered Set). | | | | | | | | |
| 1 | <p>Link Number</p> <ul style="list-style-type: none"> Ports that do not support 8.0 GT/s or above: 0-255, PAD. Downstream Ports that support 8.0 GT/s or above: 0-31, PAD. Upstream Ports that support 8.0 GT/s or above: 0-255, PAD. When operating at 2.5 or 5.0 GT/s: PAD is encoded as <u>K23.7</u>. When operating at 8.0 GT/s or above: PAD is encoded as F7h. | | | | | | | | |
| 2 | <p>Lane Number within Link</p> <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: 0-31, PAD. PAD is encoded as <u>K23.7</u>. When operating at 8.0 GT/s or above: 0-31, PAD. PAD is encoded as F7h. | | | | | | | | |
| 3 | <p>N_FTS </p> <p> When Flit_Mode_Enabled is 0b:</p> <p>The number of Fast Training Sequences required by the Receiver: 0-255.  Reserved when the Flit_Mode_Enabled is variable.</p> <p> When Flit_Mode_Enabled is Set.  1b:</p> <p> Bits 5:0  Reserved</p> <p> Bits 7:6  Optical Aware Retimer Solution Present</p> <p> 10b  Optical Aware Retimer Solution Present</p> <p> Others  Reserved</p> | | | | | | | | |
| 4 | <p>Data Rate Identifier</p> <p>Bit 0 Flit Mode Supported bit</p> <p>0b Flit Mode is not supported</p> <p>1b Flit Mode is supported</p> <p>Bits 5:1 Supported Link Speeds :</p> <p>Non-Flit Mode Encodings (valid during Flit Mode negotiation or when Flit Mode is not negotiated)</p> <table> <tbody> <tr> <td>0 0001b</td> <td>Only 2.5 GT/s supported</td> </tr> <tr> <td>0 0011b</td> <td>Only 2.5 and 5.0 GT/s supported</td> </tr> <tr> <td>0 0111b</td> <td>Only 2.5, 5.0, and 8.0 GT/s supported</td> </tr> <tr> <td>0 1111b</td> <td>Only 2.5, 5.0, 8.0, and 16.0 GT/s supported</td> </tr> </tbody> </table> | 0 0001b | Only 2.5 GT/s supported | 0 0011b | Only 2.5 and 5.0 GT/s supported | 0 0111b | Only 2.5, 5.0, and 8.0 GT/s supported | 0 1111b | Only 2.5, 5.0, 8.0, and 16.0 GT/s supported |
| 0 0001b | Only 2.5 GT/s supported | | | | | | | | |
| 0 0011b | Only 2.5 and 5.0 GT/s supported | | | | | | | | |
| 0 0111b | Only 2.5, 5.0, and 8.0 GT/s supported | | | | | | | | |
| 0 1111b | Only 2.5, 5.0, 8.0, and 16.0 GT/s supported | | | | | | | | |

Base 6.4 vs Base 6.3

TS2 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description | | | | | | | | | | | | | | | | | | | | |
|---------------|--|-----------|----------|-----------|--------|-----------|----------|-----------|--------|-----------|----------|-----------|--------|-----------|----------|-----------|--------|-----------|---------------------|-----------|------------------------------|
| | <p>1 1111b Only 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s supported</p> <p>Others Reserved in Non-Flit Mode</p> <p>Additional encodings permitted after Flit Mode is negotiated by all Link Partners after the first entry to Configuration.Complete from Detect.</p> | | | | | | | | | | | | | | | | | | | | |
| | <p>1 0111b 2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s supported</p> <p>Others Reserved</p> | | | | | | | | | | | | | | | | | | | | |
| | <p>Bit 6 Autonomous Change / Selectable De-emphasis / Link Upconfigure / LOp Capability - This bit is defined for use in the following LTSSM states: Polling.Configuration , Configuration.Complete , and Recovery . In all other LTSSM states, it is Reserved.</p> | | | | | | | | | | | | | | | | | | | | |
| | <p>Bit 7 speed_change / <i>SRIS Clocking</i></p> <ul style="list-style-type: none"> • Downstream Ports: <ul style="list-style-type: none"> ◦ In Configuration State: <ul style="list-style-type: none"> ▪ When LinkUp=0b: A 1b indicates that the Link will operate in SRIS clocking; a 0b indicates either common clocking or SRNS clocking. The Downstream Port uses this bit to communicate the type of clocking to the Retimer(s), if any, as well as the Upstream Port so that the correct (Control) SKP Ordered Set frequency can be selected ▪ Else: this bit is Reserved ◦ In Recovery.RcvrCfg : speed_change ◦ In other states: Reserved • Upstream Ports: speed_change . This bit can be set to 1b only in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved | | | | | | | | | | | | | | | | | | | | |
| 5 | <p>Training Control</p> <p>Bit 0 Hot_Reset_Request bit</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0b</td> <td>Deassert</td> </tr> <tr> <td>1b</td> <td>Assert</td> </tr> </table> <p>Bit 1 Disable_Link_Request bit</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0b</td> <td>Deassert</td> </tr> <tr> <td>1b</td> <td>Assert</td> </tr> </table> <p>Bit 2 Loopback_Request bit</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0b</td> <td>Deassert</td> </tr> <tr> <td>1b</td> <td>Assert</td> </tr> </table> <p>Bit 3 Disable Scrambling bit in 2.5 GT/s and 5.0 GT/s data rates; Reserved in other data rates</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0b</td> <td>Deassert</td> </tr> <tr> <td>1b</td> <td>Assert</td> </tr> </table> <p>Bit 4 <i>Retimer Present bit</i> in 2.5 GT/s data rate. Reserved in other data rates.</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0b</td> <td>No Retimers present</td> </tr> <tr> <td>1b</td> <td>One or more Retimers present</td> </tr> </table> <p>Bit 5 <i>Two Retimers Present bit</i> in 2.5 GT/s data rate. Reserved in other data rates.</p> | 0b | Deassert | 1b | Assert | 0b | No Retimers present | 1b | One or more Retimers present |
| 0b | Deassert | | | | | | | | | | | | | | | | | | | | |
| 1b | Assert | | | | | | | | | | | | | | | | | | | | |
| 0b | Deassert | | | | | | | | | | | | | | | | | | | | |
| 1b | Assert | | | | | | | | | | | | | | | | | | | | |
| 0b | Deassert | | | | | | | | | | | | | | | | | | | | |
| 1b | Assert | | | | | | | | | | | | | | | | | | | | |
| 0b | Deassert | | | | | | | | | | | | | | | | | | | | |
| 1b | Assert | | | | | | | | | | | | | | | | | | | | |
| 0b | No Retimers present | | | | | | | | | | | | | | | | | | | | |
| 1b | One or more Retimers present | | | | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

6

TS2 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------|---|----------------|---|----------------|--|--------------|------------|--------------|--|----------------|----------|----------------|--|--------------|------------|----------------|-----------|----------------|---|------------|----------|------------|-----------|------------|-----------|------------|-----------|
| | Ports that support 16.0 GT/s data rate or higher must implement this bit. Ports that support only 8.0 GT/s data rate or lower are permitted to implement this bit. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0b | Zero or one Retimers present (or bit not implemented) | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1b | Two or more Retimers present | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 7:6 | Enhanced Link Behavior Control | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00b | <u>Full Equalization Required ,</u> <u>Modified TS1/TS2 Ordered Sets not supported.</u> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01b | <u>Equalization Bypass to Highest NRZ Rate Support</u> <u>Modified TS1/TS2 Ordered Sets not supported.</u> See § Section 4.2.4 . | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10b | <u>No Equalization Needed ,</u> A device advertising this capability must support Equalization Bypass to Highest NRZ Rate Support. See § Section 4.2.4 . <u>Modified TS1/TS2 Ordered Sets not supported</u> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11b | <u>Modified TS1/TS2 Ordered Sets supported,</u> <u>Equalization bypass options specified in Modified TS1/TS2 Ordered Sets .</u> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | These bits defined for use in Polling and Configuration when <u>LinkUp</u> =0 and 32.0 GT/s or higher data rate is supported. In all other cases, Bits 7:6 are Reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | <ul style="list-style-type: none"> • When operating at 2.5 or 5.0 GT/s: <ul style="list-style-type: none"> ◦ Standard <u>TS2</u> Ordered Sets encode this Symbol as a <u>TS2 Identifier</u>, D5.2 (45h). ◦ <u>EQ TS2</u> Ordered Sets encode this Symbol as follows: <ul style="list-style-type: none"> ▪ For Equalization at 8.0 GT/s Data Rate: <table> <tr> <td>Bit 2:0</td> <td>Receiver Preset Hint. See § Section 4.2.4.2 .</td> </tr> <tr> <td>Bit 6:3</td> <td>Transmitter Preset . See § Section 4.2.4.2 .</td> </tr> <tr> <td>Bit 7</td> <td>Set to 1b.</td> </tr> </table> ▪ For Equalization at 32.0 GT/s Data Rate: <table> <tr> <td>Bit 0</td> <td>Transmitter Precode Request . See § Section 4.2.2.5 and § Section 4.2.3.1.4 .</td> </tr> <tr> <td>Bit 2:1</td> <td>Reserved</td> </tr> <tr> <td>Bit 6:3</td> <td>Transmitter Preset . See § Section 4.2.4.2 .</td> </tr> <tr> <td>Bit 7</td> <td>Set to 1b.</td> </tr> </table> • When operating at 8.0 GT/s or higher: <table> <tr> <td>Bit 3:0</td> <td>Reserved.</td> </tr> <tr> <td>Bit 5:4</td> <td>Equalization Request Data Rate .</td> </tr> <tr> <td>00b</td> <td>8.0 GT/s</td> </tr> <tr> <td>10b</td> <td>16.0 GT/s</td> </tr> <tr> <td>01b</td> <td>32.0 GT/s</td> </tr> <tr> <td>11b</td> <td>64.0 GT/s</td> </tr> </table> | Bit 2:0 | Receiver Preset Hint. See § Section 4.2.4.2 . | Bit 6:3 | Transmitter Preset . See § Section 4.2.4.2 . | Bit 7 | Set to 1b. | Bit 0 | Transmitter Precode Request . See § Section 4.2.2.5 and § Section 4.2.3.1.4 . | Bit 2:1 | Reserved | Bit 6:3 | Transmitter Preset . See § Section 4.2.4.2 . | Bit 7 | Set to 1b. | Bit 3:0 | Reserved. | Bit 5:4 | Equalization Request Data Rate . | 00b | 8.0 GT/s | 10b | 16.0 GT/s | 01b | 32.0 GT/s | 11b | 64.0 GT/s |
| Bit 2:0 | Receiver Preset Hint. See § Section 4.2.4.2 . | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 6:3 | Transmitter Preset . See § Section 4.2.4.2 . | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 7 | Set to 1b. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 0 | Transmitter Precode Request . See § Section 4.2.2.5 and § Section 4.2.3.1.4 . | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 2:1 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 6:3 | Transmitter Preset . See § Section 4.2.4.2 . | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 7 | Set to 1b. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 3:0 | Reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit 5:4 | Equalization Request Data Rate . | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00b | 8.0 GT/s | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10b | 16.0 GT/s | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01b | 32.0 GT/s | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11b | 64.0 GT/s | | | | | | | | | | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

TS2 Ordered Set in 8b/10b and 128b/130b Encoding

| Symbol Number | Description |
|---------------|--|
| | <p>These bits are defined for use in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, they are Reserved. See § Section 4.2.4 for usage and recognize that these bits are non-sequentially encoded for purposes of backwards compatibility</p> <p>Bit 6 <i>Quiesce Guarantee</i> . This bit is defined for use in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved.</p> <p>Bit 7 <i>Request Equalization</i> . This bit is defined for use in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved.</p> |
| 7 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: TS2 Identifier. Encoded as D5.2 (45h). When operating at 8.0 GT/s or higher Data Rate: <ul style="list-style-type: none"> Standard TS2 Ordered Sets encode this Symbol as a TS2 Identifier, 45h. 128b/130b EQ TS2 Ordered Sets encode this Symbol as follows: <p>Bit 0 Transmitter Precode Request for operating at 32.0 GT/s or higher Data Rate. See § Section 4.2.2.5 . This bit is Reserved if the 128b/130b EQ TS2 is sent for equalization at data rates of 8.0 GT/s or 16.0 GT/s.</p> <p>Bit 2:1 Reserved</p> <p>Bit 6:3 128b/130b Transmitter Preset. See § Section 4.2.4.2 .</p> <p>Bit 7 Set to 1b.</p> <p>This definition is only valid in the Recovery.RcvrCfg LTSSM state when Preset values are being communicated.</p> |
| 8 - 13 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: TS2 Identifier. Encoded as D5.2 (45h). When operating at 8.0 GT/s or above: TS2 Identifier. Encoded as 45h. |
| 14-15 | <ul style="list-style-type: none"> When operating at 2.5 or 5.0 GT/s: TS2 Identifier. Encoded as D5.2 (45h). When operating at 8.0 GT/s or above: TS2 Identifier (encoded as 45h) or a DC Balance Symbol. |

Table 4-38 Modified TS1/TS2 Ordered Set (8b/10b encoding) §

| Modified TS1/TS2 Ordered Set (8b/10b encoding) | |
|--|--|
| Symbol Number | Description |
| 0 | COM (K28.5) for Symbol alignment. |
| 1 | <p>Link Number</p> <p>Downstream Ports: 0-31, PAD (K23.7).</p> <p>Upstream Ports: 0-255, PAD (K23.7).</p> |
| 2 | Lane Number within Link - 0-31, PAD. PAD is encoded as K23.7 . |

Base 6.4 vs Base 6.3

Modified TS1/TS2 Ordered Set (8b/10b encoding)

| Symbol Number | Description |
|---------------|--|
| 3 | <p>N_FTS</p> <p>When Flit_Mode_Enabled is 0b:</p> <p>The number of Fast Training Sequences required by the Receiver: 0-255.</p> <p>When Flit_Mode_Enabled is variable:</p> <p>Set: 1b</p> <p>Others: Reserved</p> <p>ECN: Base 6.3 Optical</p> |
| 4 | <p>Data Rate Identifier</p> <p>Bit 0 Flit Mode Supported bit</p> <p>0b Flit Mode is not supported</p> <p>1b Flit Mode is supported</p> <p>Bits 5:1 — Data Rates Supported</p> <ul style="list-style-type: none"> 0 0001b Only 2.5 GT/s Data Rate Supported. 0 0011b Only 2.5 and 5.0 GT/s Data Rate Supported. 0 0111b Only 2.5, 5.0, and 8.0 GT/s Data Rate Supported. 0 1111b Only 2.5, 5.0, 8.0, and 16.0 GT/s Data Rate Supported. 1 1111b Only 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s Data Rate Supported. <p>Others Reserved in Non-Flit Mode</p> <p>Additional encodings permitted after Flit Mode is negotiated by all Link Partners after the first entry to Configuration.Complete from Detect :</p> <ul style="list-style-type: none"> 1 0111b 2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s supported Others Reserved <p>Bit 6 Link Upconfigure / Lop Capability – This bit is defined for use in Configuration.Complete in Modified TS2 Ordered Sets. In all other LTSSM states, it is Reserved.</p> <p>Bit 7 SRIS Clocking – When LinkUp =0b: A 1b indicates that the Link will operate in SRIS clocking; a 0b indicates either common clocking or SRNS clocking. The Downstream Port uses this bit to communicate the type of clocking to the Retimer(s), if any, as well as the Upstream Port so that the correct (Control) SKP Ordered Set frequency can be selected</p> <p>Else: this bit is Reserved</p> |
| 5 | <p>Training / Equalization Control</p> <p>Bit 0 Equalization Bypass to Highest NRZ Rate Support . See § Section 4.2.4</p> <p>Bit 1 No Equalization Needed bit . See § Section 4.2.4</p> <p>Bit 3:2 Reserved</p> <p>Bit 4 Retimer Present bit</p> <p>0b No Retimers present</p> |

Base 6.4 vs Base 6.3

Modified TS1/TS2 Ordered Set (8b/10b encoding)

| Symbol Number | Description | |
|---------------|--|---|
| | 1b | One Retimer is present |
| | Bit 5 | <u>Two Retimers Present bit</u> |
| | 0b | Zero or one Retimers present |
| | 1b | Two or more Retimers present |
| | Bit 6 | 1b |
| | Bit 7 | 1b |
| 6 | For Modified TS1 : TS1 Identifier, encoded as D10.2 For Modified TS2 : TS2 Identifier, encoded as D5.2 | |
| 7 | For Modified TS1 : TS1 Identifier, encoded as D10.2 For Modified TS2 : TS2 Identifier, encoded as D5.2 | |
| 8-9 | Bits 2:0 | Modified TS Usage |
| | 000b | PCIe protocol only |
| | 001b | PCIe protocol only with vendor defined Training Set Messages |
| | 010b | Alternate Protocol Negotiation |
| | Others | Reserved |
| | The values advertised in these bits must be consistent with the <u>Modified TS Usage Mode Selected</u> field of the <u>32.0 GT/s Control register</u> and the capabilities of the device. These are bits[2:0] of Symbol 8. | |
| | Bits 15:3 | Modified TS Information 1 |
| | If <u>Modified TS Usage</u> = 001b or 010b; else Reserved. | |
| 10-11 | Training Set Message Vendor ID if <u>Modified TS Usage</u> = 001b. Alternate Protocol Vendor ID if <u>Modified TS Usage</u> = 010b. Reserved for other cases. | |
| 12-14 | If <u>Modified TS Usage</u> = 001b or 010b, Modified TS Information 2 Else, Reserved | |
| 15 | Bit-wise even parity of Symbols 4 through 14. Symbol 15 = Symbol 4 ^ Symbol 5 ^ ... Symbol 14 | |

Fields in the Modified TS1/TS2 Ordered Sets that extend over multiple Symbols use the little endian format using all the bits over those multiple Symbols. For example, Symbols 8 and 9 of the Modified TS1/TS2 comprise 16 bits. The Modified TS Usage field goes in bits [2:0] of Symbol 8 with the bit 0 of Modified TS Usage field placed in bit 0 of Symbol 8, bit 1 of Modified TS Usage field placed in bit 1 of Symbol 8, and bit 2 of Modified TS Usage field placed in bit 2 of Symbol 8.

Similarly, bit 12 of the 13 bits of Modified TS Information 1 field is placed in bit 7 of Symbol 9 whereas bit 0 of Modified TS Information 1 is placed in bit 3 of Symbol 8.

Table 4-39 TS1/TS2 Ordered Set with 1b/1b Encoding §

| TS1/TS2 Ordered Set with 1b/1b Encoding | | | | | | | | | | | |
|---|--|-------------|----------|-------------|-----------|-------------|-----------|-------------|-----------|---------------|----------|
| Symbol Numbers | Description | | | | | | | | | | |
| | TS1/TS2 Identifier - Unscrambled | | | | | | | | | | |
| 0, 8 | <ul style="list-style-type: none"> Encoded as 1Bh for TS1 Encoded as 39h for TS2 | | | | | | | | | | |
| 1, 9 | <ul style="list-style-type: none"> Link Number in Configuration , Hot Reset , or Recovery.RcvrCfg state - Scrambled <ul style="list-style-type: none"> 0-31, PAD (F7h) As a Receiver in Recovery.Idle , this Byte is only used to check for PAD (F7h) Equalization Byte 0 in Recovery and Loopback for TS1 Ordered Set - Scrambled <ul style="list-style-type: none"> Bits 1:0 Equalization Control (EC) - These bits are defined for use in Recovery.Equalization and Loopback.Entry . In all other Recovery substates these bits are 00b. In all other Loopback states, these bits are Reserved. Bit 2 Reset EIEOS Interval Count - This bit is defined for use in Recovery.Equalization . In all other Recovery substates, and in Loopback , this bit is Reserved. Bits 6:3 Transmitter Preset . These bits are defined for use in Recovery and Loopback.Entry – See § Section 4.2.4.2 , § Section 4.2.7.4 , and § Section 4.2.7.10.1 . In all other states these bits are Reserved. Bit 7 Use Preset / Equalization Redo - This bit is defined for use in Recovery.Equalization , Recovery.RcvrLock , and Loopback.Entry . See § Section 4.2.7.4.2 , § Section 4.2.7.4.1 , and § Section 4.2.7.10.1 . In all other Recovery substates and Loopback substates, it is Reserved. Equalization Byte 0 in Recovery for TS2 Ordered Set - Scrambled <ul style="list-style-type: none"> Bit 0 0b Bits 2:1 Reserved Bits 5:3 Equalization Request Data Rate <table> <tr> <td>000b</td> <td>8.0 GT/s</td> </tr> <tr> <td>001b</td> <td>16.0 GT/s</td> </tr> <tr> <td>010b</td> <td>32.0 GT/s</td> </tr> <tr> <td>011b</td> <td>64.0 GT/s</td> </tr> <tr> <td>Others</td> <td>Reserved</td> </tr> </table> See § Section 4.2.4 for usage. Bit 6 Quiesce Guarantee Bit 7 Request Equalization • Reserved in other states - Scrambled | 000b | 8.0 GT/s | 001b | 16.0 GT/s | 010b | 32.0 GT/s | 011b | 64.0 GT/s | Others | Reserved |
| 000b | 8.0 GT/s | | | | | | | | | | |
| 001b | 16.0 GT/s | | | | | | | | | | |
| 010b | 32.0 GT/s | | | | | | | | | | |
| 011b | 64.0 GT/s | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | |
| 2, 10 | <ul style="list-style-type: none"> Lane Number in Configuration or Hot Reset state - Scrambled <ul style="list-style-type: none"> PAD is encoded as F7h | | | | | | | | | | |

Base 6.4 vs Base 6.3

TS1/TS2 Ordered Set with 1b/1b Encoding

| Symbol Numbers | Description | | | | | | | | | | | | | | | | | | | | |
|-------------------|---|--------------|----------|-----------------|-----------------------------|----------------|-------------------------|----------------|---------------------------------|----------------|---------------------------------------|----------------|---|----------------|---|----------------|--|---------------|----------|--------------|---|
| | <ul style="list-style-type: none"> As a Receiver in <u>Recovery.Idle</u>, this Byte is only used to check for PAD (F7h) Equalization Byte 1 in <u>Recovery</u> and <u>Loopback.Entry</u> - Scrambled <ul style="list-style-type: none"> Bits 5:0 Cursor C_0 for the current data rate of operation. Bit 6 Transmitter Precoding On (Recovery only, Reserved in Loopback.Entry) Bit 7 Retimer Equalization Extend bit (Recovery only, Reserved in Loopback.Entry) Reserved in other states - Scrambled | | | | | | | | | | | | | | | | | | | | |
| 3, 11 | <ul style="list-style-type: none"> Equalization Byte 2 in <u>Recovery</u> and <u>Loopback.Entry</u> for TS1 Ordered Sets , Reserved for TS2 Ordered Sets - Scrambled <ul style="list-style-type: none"> Bits 3:0 First Pre-cursor Coefficient [3:0] (C_{-1}) for the current data rate of operation Bits 6:4 Second Pre-cursor Coefficient [2:0] (C_{-2}) for the current data rate of operation. Bit 7 Reject Coefficient Values bit - This bit can only be set to 1b in specific phases of the <u>Recovery.Equalization</u> LTSSM state. See § <u>Section 4.2.7.4.2</u> . In all other <u>Recovery</u> substates it must be set to 0b. Reserved in other states - Scrambled | | | | | | | | | | | | | | | | | | | | |
| 4, 12 | <ul style="list-style-type: none"> Equalization Byte 3 in <u>Recovery</u> and <u>Loopback.Entry</u> for TS1 Ordered Sets , Reserved for TS2 Ordered Sets - Scrambled <ul style="list-style-type: none"> Bits 4:0 Post-cursor Coefficient [4:0] (C_{+1}) for the current data rate of operation. Bits 7:5 Reserved Reserved in other states - Scrambled | | | | | | | | | | | | | | | | | | | | |
| 5, 13 | <p>Data Rate Identifier - Scrambled</p> <table> <tbody> <tr> <td>Bit 0</td><td>Reserved</td></tr> <tr> <td>Bits 5:1</td><td>Data Rates Supported</td></tr> <tr> <td>0 0001b</td><td>Only 2.5 GT/s supported</td></tr> <tr> <td>0 0011b</td><td>Only 2.5 and 5.0 GT/s supported</td></tr> <tr> <td>0 0111b</td><td>Only 2.5, 5.0, and 8.0 GT/s supported</td></tr> <tr> <td>0 1111b</td><td>Only 2.5, 5.0, 8.0, and 16.0 GT/s supported</td></tr> <tr> <td>1 1111b</td><td>Only 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s supported</td></tr> <tr> <td>1 0111b</td><td>2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s supported</td></tr> <tr> <td>Others</td><td>Reserved</td></tr> <tr> <td>Bit 6</td><td><u>Autonomous Change / Selectable De-emphasis</u></td></tr> </tbody> </table> <p>Downstream Ports: This bit is defined for use in the following LTSSM states: <u>Configuration</u>, <u>Linkwidth.Start</u> and <u>Loopback.Entry</u> for TS1 Ordered Sets, and <u>Recovery</u> for TS2 Ordered Sets. In all other states, it is Reserved.</p> <p>Upstream Ports: This bit is defined for use in the following LTSSM states: <u>Configuration</u>, <u>Recovery</u> and <u>Loopback.Entry</u>. In all other states, it is Reserved.</p> | Bit 0 | Reserved | Bits 5:1 | Data Rates Supported | 0 0001b | Only 2.5 GT/s supported | 0 0011b | Only 2.5 and 5.0 GT/s supported | 0 0111b | Only 2.5, 5.0, and 8.0 GT/s supported | 0 1111b | Only 2.5, 5.0, 8.0, and 16.0 GT/s supported | 1 1111b | Only 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s supported | 1 0111b | 2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s supported | Others | Reserved | Bit 6 | <u>Autonomous Change / Selectable De-emphasis</u> |
| Bit 0 | Reserved | | | | | | | | | | | | | | | | | | | | |
| Bits 5:1 | Data Rates Supported | | | | | | | | | | | | | | | | | | | | |
| 0 0001b | Only 2.5 GT/s supported | | | | | | | | | | | | | | | | | | | | |
| 0 0011b | Only 2.5 and 5.0 GT/s supported | | | | | | | | | | | | | | | | | | | | |
| 0 0111b | Only 2.5, 5.0, and 8.0 GT/s supported | | | | | | | | | | | | | | | | | | | | |
| 0 1111b | Only 2.5, 5.0, 8.0, and 16.0 GT/s supported | | | | | | | | | | | | | | | | | | | | |
| 1 1111b | Only 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s supported | | | | | | | | | | | | | | | | | | | | |
| 1 0111b | 2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s supported | | | | | | | | | | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | | | | | | | | | | |
| Bit 6 | <u>Autonomous Change / Selectable De-emphasis</u> | | | | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

TS1/TS2 Ordered Set with 1b/1b Encoding

| Symbol Numbers | Description | |
|----------------|---|---|
| | Bit 7 | speed_change - This bit can be set to 1b only in the Recovery LTSSM state. In all other LTSSM states, it is Reserved. |
| | Training Control | Used in Recovery.RcvrCfg to Disable , Hot Reset , or Loopback . Reserved for TS2 Ordered Sets in Configuration - Scrambled |
| | Bits 3:0 | <p>0000b Deassert</p> <p>0001b Assert Hot_Reset_Request</p> <p>0010b Assert Disable_Link_Request</p> <p>0100b Assert Loopback_Request - the Follower Port at Receiver (A or F) loops back to its Transmitter</p> <p>0101b Assert Loopback_Request – the Pseudo-Port Receiver B or C loops back to its Transmitter, depending on which Port is the Loopback Lead (Follower Port still loops back except the Pseudo-Port that is acting as the Follower does not forward the bits)</p> |
| 6, 14 | | What does "the bits" mean? |
| | 0110b | <p>Assert Loopback_Request – the Pseudo-Port Receiver D or E loops back to its Transmitter, depending on which Port is the Loopback Lead (Follower Port still loops back except the Pseudo-Port that is acting as the Follower does not forward the bits)</p> <p>What does "the bits" mean?</p> |
| | 1000b | Assert Compliance_Receive_Request |
| | 1100b | Assert Loopback_Request and Compliance_Receive_Request |
| | Others | Reserved |
| | Bits 7:4 | Reserved |
| 7, 15 | <p>If DC Balance needs adjustment at the start of the TS1 or TS2 :</p> <p>DC Balance Symbol - Unscrambled</p> <p>else:</p> <p>Byte level even parity over Symbols 0-6 (or 8-14) - Scrambled</p> <p>Symbol 7 = Symbol 0 ^ Symbol 1 ^ ... Symbol 6</p> <p>Symbol 15 = Symbol 8 ^ Symbol 9 ^ ... Symbol 14</p> | |

Table 4-40 TS0 Ordered Set §

| TS0 Ordered Set | | |
|-----------------|---|---|
| Symbol Numbers | Description | |
| | TS0 Identifier - Unscrambled 33h | |
| 0, 8 | Bits 7,5,3,1 | 0101b |
| | Bits 6,4,2,0 | Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling . |

| | | <u>TS0 Ordered Set</u> | | | | | | | | | | | | | | | | |
|-------------------------|---|-------------------------|-----------------------------------|-------------------------|--|---------------|---|------------|--|-----------|----------------------------------|-----------|---------------------------|-----------|------------------|-----------|------------|--|
| Symbol Numbers | Description | | | | | | | | | | | | | | | | | |
| | Equalization Byte 0 - Half Scrambled Symbol 1 determines the interpretation of Symbols 2 through 6. Symbol 9 determines the interpretation of Symbols 10 through 14. | | | | | | | | | | | | | | | | | |
| 1, 9 | <p>Bits 3,1 Equalization Control (EC)</p> <table> <tr> <td>00b</td><td>Phase 0 (used during Phase 0 & 1)</td></tr> <tr> <td>01b</td><td>Phase 1 (used during Phase 0 & 1)</td></tr> <tr> <td>10b</td><td>Phase 2 (used during Phase 2 by Upstream Lane and by Downstream Lane only when initially requesting Upstream Lane to move to Phase 2)</td></tr> <tr> <td>11b</td><td>Phase 3 (only set by Upstream Lane initially when requesting Downstream Lane to move to Phase 3)</td></tr> </table> <p>The proper values must be sent during the corresponding phase of <u>Recovery.Equalization</u>.</p> <p>Bit 5 Reset EEOS Interval Count - This bit is defined for use in <u>Recovery.Equalization Phase 2</u>. Reserved in all other states.</p> <table> <tr> <td>0b</td><td>Do not reset EEOS Interval Count</td></tr> <tr> <td>1b</td><td>Reset EEOS Interval Count</td></tr> </table> <p>Bit 7 Use Preset - This bit is defined for use in <u>Recovery.Equalization Phase 2 & 3</u>. Use Preset is 0b in <u>Recovery.Equalization Phase 3</u> for Upstream Ports and in <u>Recovery.Equalization Phase 2</u> for Downstream Ports. Reserved in all other states.</p> <table> <tr> <td>0b</td><td>Use Coefficients</td></tr> <tr> <td>1b</td><td>Use Preset</td></tr> </table> <p>Bits 6,4,2,0 Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling.</p> | 00b | Phase 0 (used during Phase 0 & 1) | 01b | Phase 1 (used during Phase 0 & 1) | 10b | Phase 2 (used during Phase 2 by Upstream Lane and by Downstream Lane only when initially requesting Upstream Lane to move to Phase 2) | 11b | Phase 3 (only set by Upstream Lane initially when requesting Downstream Lane to move to Phase 3) | 0b | Do not reset EEOS Interval Count | 1b | Reset EEOS Interval Count | 0b | Use Coefficients | 1b | Use Preset | |
| 00b | Phase 0 (used during Phase 0 & 1) | | | | | | | | | | | | | | | | | |
| 01b | Phase 1 (used during Phase 0 & 1) | | | | | | | | | | | | | | | | | |
| 10b | Phase 2 (used during Phase 2 by Upstream Lane and by Downstream Lane only when initially requesting Upstream Lane to move to Phase 2) | | | | | | | | | | | | | | | | | |
| 11b | Phase 3 (only set by Upstream Lane initially when requesting Downstream Lane to move to Phase 3) | | | | | | | | | | | | | | | | | |
| 0b | Do not reset EEOS Interval Count | | | | | | | | | | | | | | | | | |
| 1b | Reset EEOS Interval Count | | | | | | | | | | | | | | | | | |
| 0b | Use Coefficients | | | | | | | | | | | | | | | | | |
| 1b | Use Preset | | | | | | | | | | | | | | | | | |
| 2, 10 | <p>Equalization Byte 1 - Half Scrambled</p> <p>For the current data rate of operation. Interpretation depends on the EC field of Symbol 1 (or 9)</p> <p>Bits 7,5,3,1</p> <table> <tr> <td>Phases 0 & 1</td><td>FS[3:0]</td></tr> <tr> <td>Phases 2 & 3</td><td>First Pre-Cursor Coefficient [3:0] (C_{-1}) when Use Preset field of symbol 1, 9 is 0b</td></tr> <tr> <td>Others</td><td>Reserved</td></tr> </table> <p>Bits 6,4,2,0 Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling.</p> | Phases 0 & 1 | FS[3:0] | Phases 2 & 3 | First Pre-Cursor Coefficient [3:0] ($ C_{-1} $) when Use Preset field of symbol 1, 9 is 0b | Others | Reserved | | | | | | | | | | | |
| Phases 0 & 1 | FS[3:0] | | | | | | | | | | | | | | | | | |
| Phases 2 & 3 | First Pre-Cursor Coefficient [3:0] ($ C_{-1} $) when Use Preset field of symbol 1, 9 is 0b | | | | | | | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | | | | | | | |
| 3, 11 | <p>Equalization Byte 2 - Half Scrambled</p> <p>For the current data rate of operation. Interpretation depends on the EC field of Symbol 1 (or 9)</p> <p>Bits 3,1</p> <table> <tr> <td>Phases 0 & 1</td><td>FS[5:4]</td></tr> <tr> <td>Phases 2 & 3</td><td>Post-Cursor Coefficient [1:0] (C_{+1}) when Use Preset field of symbol 1, 9 is 0b</td></tr> <tr> <td>Others</td><td>Reserved</td></tr> </table> | Phases 0 & 1 | FS[5:4] | Phases 2 & 3 | Post-Cursor Coefficient [1:0] ($ C_{+1} $) when Use Preset field of symbol 1, 9 is 0b | Others | Reserved | | | | | | | | | | | |
| Phases 0 & 1 | FS[5:4] | | | | | | | | | | | | | | | | | |
| Phases 2 & 3 | Post-Cursor Coefficient [1:0] ($ C_{+1} $) when Use Preset field of symbol 1, 9 is 0b | | | | | | | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| <u>TS0 Ordered Set</u> | | |
|---|---------------------|---|
| Symbol Numbers | Description | |
| | Bits 7,5 | Phases 0 & 1 LF[1:0] Phases 2 & 3 Post-Cursor Coefficient [3:2] ($ C_{+1} $) when Use Preset field of symbol 1, 9 is 0b Others Reserved |
| | Bits 6,4,2,0 | Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling . |
| Equalization Byte 3 - Half Scrambled | | |
| | | For the current data rate of operation. Interpretation depends on the EC field of Symbol 1 (or 9) |
| 4, 12 | Bit 1 | Phases 0 & 1 LF[2] Phases 2 & 3 Post-Cursor Coefficient [4] ($ C_{+1} $) when Use Preset field of symbol 1, 9 is 0b Others Reserved |
| | Bits 7,5,3 | Phases 0 & 1 LF[5:3] Phases 2 & 3 Second Pre-Cursor Coefficient [2:0] ($ C_{-2} $) when Use Preset field of symbol 1, 9 is 0b Others Reserved |
| | Bits 6,4,2,0 | Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling . |
| | | |
| Equalization Byte 4 - Half Scrambled | | |
| | | For the current data rate of operation. Interpretation depends on the EC and Use Preset fields of Symbol 1 (or 9) |
| 5, 13 | Bit 7,5,3,1 | Phases 0 & 1 Transmitter Preset [3:0] Phases 2 & 3 If Use Preset is 1b, Transmitter Preset [3:0], Else Cursor [3:0] ($ C_0 $) Others Reserved |
| | Bits 6,4,2,0 | Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling . |
| | | |
| Equalization Byte 5 - Half Scrambled | | |
| | | For the current data rate of operation. Interpretation depends on the EC field of Symbol 1 (or 9) |
| 6, 14 | Bits 3,1 | Phases 2 & 3 Cursor [5:4] ($ C_0 $) when Use Preset field of symbol 1, 9 is 0b Others Reserved |
| | Bit 5 | Retimer Equalization Extend |
| | Bit 7 | Reserved |
| | Bits 6,4,2,0 | Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling . |

| <u>TS0 Ordered Set</u> | |
|------------------------|---|
| Symbol Numbers | Description |
| 7, 15 | <p>If DC Balance adjustment needed at the start of the TS0 :</p> <p>00h or FFh – Unscrambled</p> <p>Else:</p> <p>Bits 7,5,3,1 Byte level even parity - Half Scrambled Symbol 7 = Symbol 0 ^ Symbol 1 ^ ... Symbol 6 Symbol 15 = Symbol 8 ^ Symbol 9 ^ ... Symbol 14</p> <p>Bits 6,4,2,0 Prior to Half Scrambling the bit values are don't care, but they will be identical to bits {7, 5, 3, 1} after Half Scrambling .</p> |

4.2.5.2 Alternate Protocol Negotiation §

In addition to the decision to skip equalization, alternate protocols are permitted to be negotiated during the Configuration.Lanenum.Wait , Configuration.Lanenum.Accept , and Configuration.Complete substates, while LinkUp =0b, through the exchange of Modified TS1/TS2 Ordered sets in the 8b/10b encoding. It is strongly recommended that a data rate of 32.0 GT/s or higher is advertised in all three substates throughout the entire alternate protocol negotiation procedure.

Alternate protocol(s) are permitted to be supported with PCIe PHY in 128b/130b or 1b/1b encodings. An alternate protocol is defined to be a non-PCIe protocol using the PCIe PHY layer. One may choose to run PCIe protocol in addition to one or multiple alternate protocols in the alternate protocol mode. The Ordered Set blocks are used as-is, along with the rules governing SKP Ordered Set insertion and the transition between Ordered Set and Data Blocks. The contents of the Data Blocks, however, may be modified according to the rules of the alternate protocol.

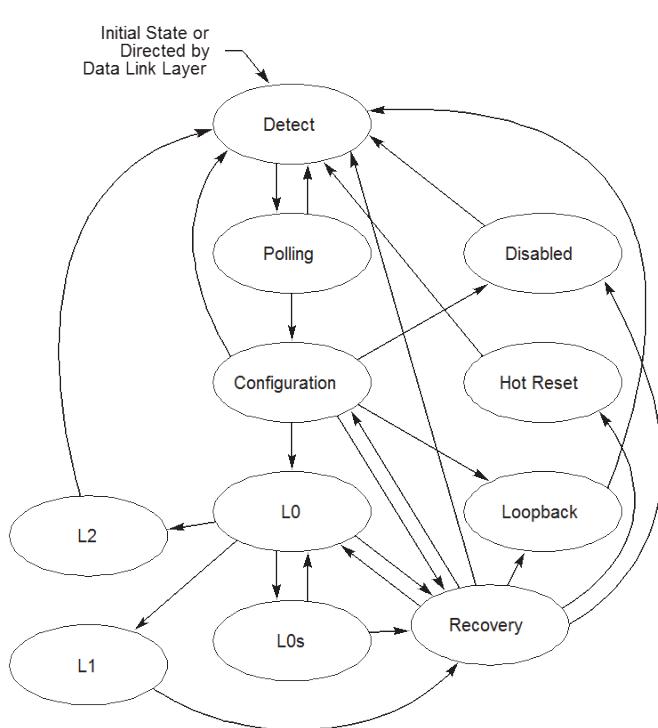
IMPLEMENTATION NOTE: ALTERNATE PROTOCOLS SHOULD HAVE AN EDS TOKEN EQUIVALENT §

The EDS Token is used in PCI Express to indicate a switch from Data Blocks to Ordered Set blocks. This additional "redundant" information ensures that a random bit error in the 2 bit block header isn't incorrectly interpreted as the end of a data stream. This is one mechanism used by PCI Express to accomplish an undetected data error Hamming Distance of 4.

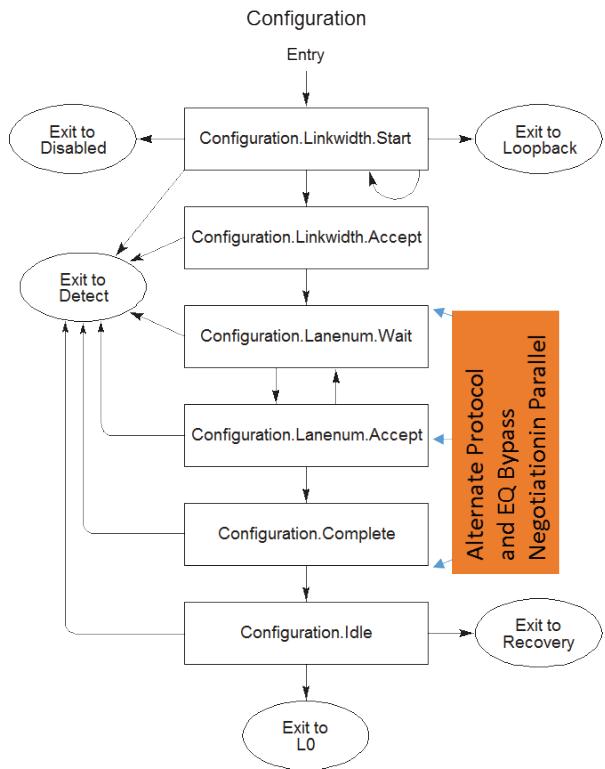
Alternate protocols should have an equivalent mechanism.

The following diagram represents the states where alternate protocol and equalization bypass negotiation occurs:

Base 6.4 vs Base 6.3



(PCIe Link Training State Machine LTSSM)



(Modified Training Sets in Config State of LTSSM to negotiate Protocol)

Figure 4-64 Alternate Protocol Negotiation and Equalization Bypass LTSSM States §

Downstream Ports manage Alternate Protocol Negotiation and Training Set Messages based on the value of the Modified TS Usage Mode Selected field when the Port is in Configuration.Lanenum.Wait, Configuration.Lanenum.Accept, and Configuration.Complete substates with LinkUp = 0.

Upstream Ports must respond to unsupported Modified TS Usage values by transmitting Modified TS Usage 000b.

If Modified TS Usage Mode Selected is:

000b

No Alternate Protocol Negotiation or Training Set Message occurs. The link will operate as a PCI Express Link.

001b

Training Set Messages are enabled. Modified TS Information 1 and Modified TS Information 2 fields carry the vendor specific messages defined by the Training Set Message Vendor ID field.

010b

Alternate Protocol Negotiation is enabled. Modified TS Information 1 and Modified TS Information 2 fields carry the alternate protocol details defined by the Alternate Protocol Vendor ID field. A protocol request or response is associated with the protocol defined by the Alternate Protocol Vendor ID field.

The Alternate Protocol Negotiation Status field indicates the progress of the negotiation protocol.

others

Reserved

A Downstream Port that supports Alternate Protocol Negotiation will start the negotiation process when it first enters Configuration.Lanenum.Wait , LinkUp = 0, and Modified TS Usage Mode Selected field is 010b. Starting negotiation consists of sending Modified TS1/TS2 Ordered Sets with Modified TS Usage = 010b.

Table 4-41 Modified TS Information 1 field in Modified TS1/TS2 Ordered Sets if Modified TS Usage = 010b (Alternate Protocol) §

| Bits | Field | Description | |
|------|--|---|--|
| 4:3 | <i>Alternate Protocol Negotiation Status</i> | For Modified TS1 Ordered Sets: | |
| | | 00b | DSP Indicates a protocol request from the Downstream Port asking whether the Upstream Port supports a particular alternate protocol. |
| | | USP | Indicates that the Upstream Port does not have an answer for a protocol request yet. This occurs either when it is evaluating the protocol request or it has not received two consecutive Modified TS1s to perform the evaluation. In the former case, Alternate Protocol Vendor ID and Alternate Protocol Details reflect what it received, while Modified TS Information 2 is protocol specific. In the latter case, all 3 fields must be 0. |
| | | 01b | DSP Reserved USP Indicates that the Upstream Port does not support the requested protocol. Alternate Protocol Vendor ID and Alternate Protocol Details reflect what it received. Modified TS Information 2 must be all 0s. |
| | | 10b | DSP Reserved USP Indicates that the Upstream Port supports the requested protocol. Alternate Protocol Vendor ID and Alternate Protocol Details reflect what it received, while Modified TS Information 2 field is protocol specific. |
| | | 11b | Reserved |
| | | For Modified TS2 Ordered Sets: | |
| | | 00b | Indicates a protocol confirmation from the Downstream Port as well as the Upstream Port. Behavior is undefined if the Downstream Port had not earlier received status 10b for this protocol in this instance of protocol negotiation during the Modified TS1 Ordered Sets. Similarly, behavior is undefined if the Upstream Port had not earlier transmitted status 10b for this protocol in this instance of protocol negotiation during the Modified TS1 Ordered Sets. |
| | | | No protocol is selected unless the Downstream Port sends and receives a protocol confirmation in the Modified TS2 Ordered Sets. If the Downstream Port decides not to use any Alternate Protocol, it must indicate this by transmitting Modified TS2 Ordered Set with Modified TS Usage of 000b or 001b. |
| | | 01b, 10b, 11b | Reserved |
| 15:5 | <i>Alternate Protocol Details</i> | Alternate Protocol Details is Modified TS Usage = 010b. | |

If Modified TS Usage = 001b, then Modified TS Information 1 and Modified TS Information 2 contain details of the training set messages.

Alternate Protocol Negotiation must be concurrent with the Lane number negotiation. During Alternate Protocol Negotiation, the Downstream Port requests support for one or more Alternate Protocols by sending a series of Modified TS1 Ordered Sets requesting each protocol, evaluating the resulting Modified TS1 Ordered Sets , and determining the resulting protocol prior to transitioning to the Configuration.Complete substate.

Upstream Ports where Modified TS Usage Mode 2 Supported - Alternate Protocol is Set, must respond to one or more Alternate Protocol requests from the Downstream Port with Modified TS1 Ordered Sets as per § Table 4-41 . Upstream Ports where Modified TS Usage Mode 2 Supported - Alternate Protocol is Clear, must respond to Alternate Protocol requests from the Downstream Port with Modified TS1 Ordered Sets with the Modified TS Usage other than 010b.

It is permitted for a Downstream Port to fall back to PCIe protocol if it does not determine a supported alternate protocol. It is permitted for a Downstream Port to discontinue negotiation when it has determined a protocol to select (e.g., if A Downstream Port supports protocols A and B and receives a supported indication for protocol A, it is permitted that the Downstream Port chose protocol A without asking about support for protocol B). On a successful negotiation to alternate protocol, the Link moves to L0 at 2.5 GT/s, changes the data rate to the higher data rates, performing equalization, if needed and enters L0 at the highest data rate desired. After transmitting the SDS Ordered Set in the highest data rate after equalization has been performed, the Data Blocks will carry the alternate protocol and the Link will be under the control of the alternate protocol.

If the DSP goes through Detect , it is permitted to remember which protocols were discovered prior to Detect . However, this does not circumvent the requirement for the complete alternate protocol negotiation to be performed in order to arrive at a common protocol (as described in above).

IMPLEMENTATION NOTE: ALTERNATE PROTOCOL NEGOTIATION BEFORE PCIE BASE 6.2 §

Some pre-6.2 USP may support an alternate protocol, in addition to PCIe, but not implement the one advanced protocol at a time mechanism. There may also be cases where the 1 msec wait after the matching Lane number in Configuration.Lanenum.Wait / Configuration.Lanenum.Accept sub-states may cause an early transition before arriving at a common alternate protocol by both Ports. In these cases, it is recommended that the DSP either uses an implementation specific means to start with the common supported alternate protocol or to go through Detect and negotiate with a new set of alternate protocols it did not try in the prior unsuccessful attempts.

4.2.5.3 Electrical Idle Sequences (EIOS and EIEOS) §

Before a Transmitter enters Electrical Idle, it must always send an Electrical Idle Ordered Set Sequence (EIOSQ), unless otherwise specified. An **Electrical Idle Ordered Set Sequence (EIOSQ)** is defined as one EIOS if the current Data Rate is 2.5 GT/s, 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, or 64.0 GT/s Data Rate, or two consecutive EIOSs if the current Data Rate is 5.0 GT/s.

When using 8b/10b encoding, an EIOS is a K28.5 (COM) followed by three K28.3 (IDL) Symbols. Transmitters must transmit all Symbols of an EIOS . An EIOS is received when the COM and two of the three IDL Symbols are received. When using 128b/130b encoding, an EIOS is an Ordered Set block, as defined in § Table 4-43 . When using 1b/1b encoding, an EIOS is an Ordered Set block, as defined in § Table 4-44 . Transmitters must transmit all Symbols of an EIOS if additional EIOSs are to be transmitted following it. Transmitters must transmit Symbols 0-13 of an EIOS , but are permitted to terminate the EIOS anywhere in Symbols 14 or 15, when transitioning to Electrical Idle after it. An EIOS is considered received when Symbols 0-3 of an Ordered Set Block match the definition of an EIOS if the data rate is less than 64.0 GT/s. At 64.0 GT/s, the rules governing receipt of an EIOS appears in § Section 4.2.3.1.5 .

IMPLEMENTATION NOTE: TRUNCATION OF EIOS ORDERED SET §

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

After transmitting the last Symbol of the last Electrical Idle Ordered Set , the Transmitter must be in a valid Electrical Idle state as specified by $T_{TX-IDLE-SET-TO-IDLE}$ (see § Table 8-7).

Table 4-42 Electrical Idle Ordered Set (EIOS) for 2.5 GT/s and 5.0 GT/s Data Rates §

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

Table 4-43 Electrical Idle Ordered Set (EIOS) for 128b/130b Encoding §

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

Table 4-44 Electrical Idle Ordered Set (EIOS) for 1b/1b Encoding §

| Symbol Numbers | Value | Description |
|---------------------------|-------|-----------------------------|
| 0, 2, 4, 6, 8, 10, 12, 14 | 0Fh | EIOS Identifier and Payload |
| 1, 3, 5, 7, 9, 11, 13, 15 | F0h | EIOS Identifier and Payload |

Table 4-45 Electrical Idle Exit Ordered Set (EIEOS) for 5.0 GT/s Data Rate §

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

Notes:

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

Table 4-46 Electrical Idle Exit Ordered Set (EIEOS) for 8.0 GT/s Data Rate §

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

Table 4-47 Electrical Idle Exit Ordered Set (EIEOS) for 16.0 GT/s Data Rate §

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

Table 4-48 Electrical Idle Exit Ordered Set (EIEOS) for 32.0 GT/s Data Rate §

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

Table 4-49 Electrical Idle Exit Ordered Set (EIEOS) for 64.0 GT/s Data Rate §

| Symbol Numbers | Value | Description |
|----------------|-------|---------------------------|
| 0 - 7 | 00h | Voltage level 0 for 32 UI |
| 8 - 15 | FFh | Voltage level 3 for 32 UI |

Base 6.4 vs Base 6.3

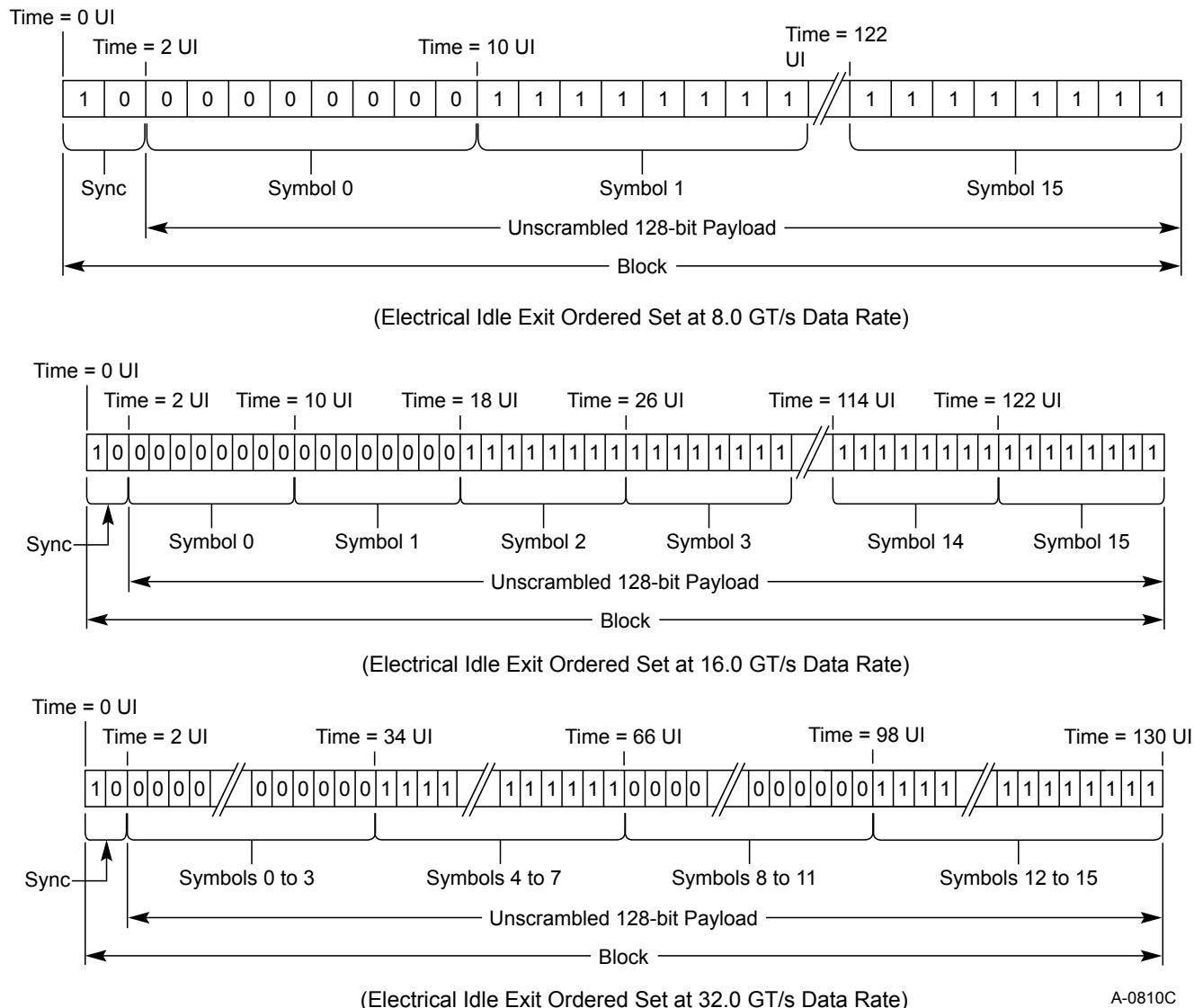


Figure 4-65 Electrical Idle Exit Ordered Set for 8.0 GT/s to 32.0 GT/s Data Rates (EIEOS) §

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

An ***Electrical Idle Exit Ordered Set Sequence (EIEOSQ)*** comprises of two consecutive EIEOS for the Data Rate of 32.0 GT/s and one EIEOS for 5.0 GT/s, 8.0 GT/s, and 16.0 GT/s. The two EIEOS at 32.0 GT/s must be back to back and uninterrupted in order to be considered consecutive and form an **EIEOSQ**. Irrespective of the length of the **EIEOSQ**, block alignment still occurs on an EIEOS.

At the Data Rate of 64.0 GT/s, an EIEOSQ is defined as follows:

- On entry to Recovery.RcvrLock either from Recovery.Speed or from L1 , until either the Receiver detects exit Electrical Idle on all Lanes or it receives two consecutive valid TS1 Ordered Sets on any Lane:
 - Four consecutive EIEOS , uninterrupted by any other Ordered Set including the Control SKP Ordered Set
- While increasing Link Width with L0p till the Receiver detects exit Electrical Idle on all Lanes that need to be activated or receives two consecutive valid TS1 Ordered Sets on any Lane that needs to be activated:
 - Four consecutive EIEOS , uninterrupted by any other Ordered Set including the Control SKP Ordered Set
- On entry to the Loopback state from electrical idle till the Receiver detects exit Electrical Idle on all Lanes that need to be activated or receives two consecutive valid TS1 Ordered Sets on any Lane:
 - Four consecutive EIEOS , uninterrupted by any other Ordered Set including the Control SKP Ordered Set
- While transmitting Compliance Patterns or Modified Compliance Patterns:
 - One EIEOS
- Else one EIEOS

When using 8b/10b encoding and operating at 5.0 GT/s, an EIEOSQ , as defined in § Table 4-45 , is transmitted in the following situations:

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

When using 128b/130b encoding, an EIEOSQ , as defined in § Table 4-46 through § Table 4-48 and § Figure 4-65 , is transmitted in the following situations:

- Before the first TS1 Ordered Set after entering the LTSSM Configuration.Linkwidth.Start substate.
- Before the first TS1 Ordered Set after entering the LTSSM Recovery.RcvrLock substate.
- Immediately following an EDS Framing Token in Non-Flit Mode when ending a Data Stream and not transmitting an EIOS and not entering the LTSSM Recovery.RcvrLock substate.
- At the scheduled Ordered Set interval to end a Data Stream in Flit Mode.
- After every 32 TS1 or TS2 Ordered Sets are transmitted in all LTSSM states which require transmission of TS1 or TS2 Ordered Sets. The TS1 / TS2 count is set to 0 when:
 - An EIEOS is transmitted.
 - The first TS2 Ordered Set is received while in the LTSSM Recovery.RcvrCfg state.
 - The first TS2 Ordered Set is received while in the LTSSM Configuration.Complete state.
 - A Downstream Port is in Phase 2 of the LTSSM Recovery.Equalization state and two consecutive TS1 Ordered Sets are received on any Lane with the Reset EIEOS Interval Count bit set.
 - An Upstream Port is in Phase 3 of the LTSSM Recovery.Equalization state and two consecutive TS1 Ordered Sets are received on any Lane with the Reset EIEOS Interval Count bit set.
- After every 65,536 TS1 Ordered Sets are transmitted in the LTSSM Recovery.Equalization state if the Reset EIEOS Interval Count bit has prevented it from being transmitted for that interval. Implementations are permitted to

satisfy this requirement by transmitting an EIEOSQ within two TS1 Ordered Sets of whenever the current scrambling LFSR matches its seed value.

- As part of an FTS Ordered Set, Compliance Pattern, or Modified Compliance Pattern as described in the relevant sections.

When using 1b/1b encoding, an EIEOSQ, as defined in § Table 4-49 is transmitted in the following situations:

- Before the first TS1 Ordered Set after entering the LTSSM Configuration.Linkwidth.Start substate.
- Before the first TS1 Ordered Set after entering the LTSSM Recovery.RcvrLock substate.
- Before the first TS0 Ordered Set after entering Recovery.Equalization substate.
- At the scheduled Ordered Set interval to indicate the end of the Data Stream.
- After every 32 TS1, TS2, or TS0 Ordered Sets are transmitted in all LTSSM states which require transmission of TS1, TS2, or TS0 Ordered Sets. The TS1/TS2/TS0 count is set to 0 when:
 - An EIEOS is transmitted.
 - The first TS2 Ordered Set is received while in the LTSSM Recovery.RcvrCfg state.
 - The first TS2 Ordered Set is received while in the LTSSM Configuration.Complete state.
 - The first TS1 Ordered Set is received while in the LTSSM Recovery.Equalization state if the TS1 Ordered Set is received after TS0 Ordered Sets.
 - A Downstream Port is in Phase 2 of the LTSSM Recovery.Equalization state and two consecutive TS0 Ordered Sets are received on any Lane with the Reset EIEOS Interval Count bit set.
 - An Upstream Port is in Phase 3 of the LTSSM Recovery.Equalization state and two consecutive TS1 Ordered Sets are received on any Lane with the Reset EIEOS Interval Count bit set.
- After every 65,536 TS1 Ordered Sets are transmitted in the LTSSM Recovery.Equalization state if the Reset EIEOS Interval Count bit has prevented it from being transmitted for that interval. Implementations are permitted to satisfy this requirement by transmitting an EIEOSQ within two TS1 Ordered Sets of whenever the current scrambling LFSR matches its seed value.
- As part of a Compliance Pattern, or Modified Compliance Pattern as described in the relevant sections.

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

While operating in speeds other than 2.5 GT/s, an implementation is permitted to not rely on the output of the Electrical Idle detection circuitry except when receiving the EIEOS during certain LTSSM states or during the receipt of the FTS prepended by the four consecutive EIE Symbols (see § Section 4.2.5.6) at the Receiver during Rx L0s or the Modified Compliance Pattern in Polling.Compliance when the circuitry is required to signal an exit from Electrical Idle.

4.2.5.4 Inferring Electrical Idle §

A device is permitted in all speeds of operation to infer Electrical Idle instead of detecting Electrical Idle using analog circuitry. § Table 4-50 summarizes the conditions to infer Electrical Idle in the various substates.

Table 4-50 Electrical Idle Inference Conditions §

| State | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s and higher data rates |
|--|---|--|--|
| L0 | Absence of at least one of: <ul style="list-style-type: none"> • an UpdateFC DLLP, • an <u>Optimized_Update_FC</u> (in Flit Mode), or • a SKP Ordered Set in a 128 µs window | | |
| <u>Recovery.RcvrCfg</u> | Absence of a <u>TS1</u> or <u>TS2</u> Ordered Set in a 1280 UI interval | | |
| <u>Recovery.Speed when successful_speed_negotiation = 1b</u> | Absence of a <u>TS1</u> or <u>TS2</u> Ordered Set in a 1280 UI interval | | Absence of a <u>TS1</u> or <u>TS2</u> Ordered Set in a 4 ms window |
| <u>Recovery.Speed when successful_speed_negotiation = 0b</u> | Absence of an exit from Electrical Idle in a 2000 UI interval | Absence of an exit from Electrical Idle in a 16000 UI interval | |
| <u>Loopback.Active</u> (as Follower) | Absence of an exit from Electrical Idle in a 128 µs window | N/A | N/A |

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

IMPLEMENTATION NOTE: INFERENCE OF ELECTRICAL IDLE §

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

4.2.5.5 Lane Polarity Inversion §

During the training sequence in Polling, the Receiver looks at Symbols 6-15 of the TS1 and TS2 Ordered Sets as the indicator of Lane polarity inversion (D+ and D- are swapped). If Lane polarity inversion occurs, the TS1 Symbols 6-15

received will be D21.5 as opposed to the expected D10.2. Similarly, if Lane polarity inversion occurs, Symbols 6-15 of the TS2 Ordered Set will be D26.5 as opposed to the expected D5.2. This provides the clear indication of Lane polarity inversion.

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

4.2.5.6 Fast Training Sequence (FTS) §

Fast Training Sequence (FTS) is the mechanism that is used for bit and Symbol lock when transitioning from L0s to L0. The FTS is used by the Receiver to detect the exit from Electrical Idle and align the Receiver's bit and Symbol receive circuitry to the incoming data. Refer to § Section 4.2.6 for a description of L0 and L0s .

- **At 2.5 GT/s and 5.0 GT/s data rates:**

A single FTS is comprised of one K28.5 (COM) Symbol followed by three K28.1 Symbols. The maximum number of FTSs (N_FTS) that a component can request is 255, providing a bit time lock of $4 * 255 * 10 * \text{UI}$. If the data rate is 5.0 GT/s, four consecutive EIE Symbols are transmitted at valid signal levels prior to transmitting the first FTS . These Symbols will help the Receiver detect exit from Electrical Idle. An implementation that does not guarantee proper signaling levels for up to the allowable time on the Transmitter pins (see § Section 4.2.5.6) since exiting Electrical Idle condition is required to prepend its first FTS by extra EIE Symbols so that the Receiver can receive at least four EIE Symbols at valid signal levels. Implementations must not transmit more than eight EIE Symbols prior to transmitting the first FTS . A component is permitted to advertise different N_FTS rates at different speeds. At 5.0 GT/s, a component may choose to advertise an appropriate N_FTS number considering that it will receive the four EIE Symbols. 4096 FTSs must be sent when the Extended Synch bit is Set in order to provide external Link monitoring tools with enough time to achieve bit and framing synchronization. SKP Ordered Sets must be scheduled and transmitted between FTSs as necessary to meet the definitions in § Section 4.2.8 with the exception that no SKP Ordered Sets can be transmitted during the first N_FTS FTSs . A single SKP Ordered Set is always sent after the last FTS is transmitted. It is permitted for this SKP Ordered Set to affect or not affect the scheduling of subsequent SKP Ordered Sets for Clock Tolerance Compensation by the Transmitter as described in § Section 4.2.8 . Note that it is possible that two SKP Ordered Sets can be transmitted back to back (one SKP Ordered Set to signify the completion of the 4096 FTSs and one scheduled and transmitted to meet the definitions described in § Section 4.2.8).

- **At 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s data rates:**

A single FTS is a 130-bit unscrambled Ordered Set Block, as shown in § Table 4-51 . The maximum number of FTSs (N_FTS) that a component can request is 255, providing a bit time lock of $130 * 255 \text{ UI}$ ($130 * 263$ or 273 UI if including the periodic EIEOS). A component is permitted to advertise different N_FTS values at different speeds. On exit from L0s , the transmitter first transmits an EIEOSQ which will help the Receiver detect exit from Electrical Idle due to its low frequency content. After that first EIEOSQ , the transmitter must send the required number of FTS (4096 when the Extended Synch bit is Set; otherwise N_FTS), with an EIEOSQ transmitted after every 32 FTS . The FTS sequence will enable the Receiver obtain bit lock (and optionally to do Block alignment). When the Extended Synch bit is Set, SKP Ordered Sets must be scheduled and transmitted between FTSs and EIEOSQ as necessary to meet the definitions in § Section 4.2.8 . The last FTS Ordered Set of the FTS sequence, if any (no FTS Ordered Sets are sent if N_FTS is equal to zero), is followed by a final EIEOSQ that will help the Receiver acquire Block alignment. Implementations are permitted to send two EIEOS back to back even at a data rate below 32.0 GT/s following the last FTS Ordered Set if the N_FTS is a multiple of 32. The EIEOS resets the scrambler in both the Transmitter as well as the Receiver. Following the final EIEOSQ , an SDS Ordered Set is transmitted to help the Receiver perform de-skew and to indicate the transition from Ordered Sets to Data Stream. After the SDS Ordered Set is transmitted, a Data Block must be transmitted.

IMPLEMENTATION NOTE: SCRAMBLING LFSR DURING FTS TRANSMISSION IN 128B/ 130B ENCODING §

Since the scrambler is reset on the last EIEOS, and none of the Ordered Set in the FTS sequence is scrambled, it does not matter whether implementations choose to advance the scrambler or not during the time FTS is received.

*Table 4-51 FTS for
8.0 GT/s and Above Data
Rates §*

| Symbol Number | Value |
|---------------|-------|
| 0 | 55h |
| 1 | 47h |
| 2 | 4Eh |
| 3 | C7h |
| 4 | CCh |
| 5 | C6h |
| 6 | C9h |
| 7 | 25h |
| 8 | 6Eh |
| 9 | ECh |
| 10 | 88h |
| 11 | 7Fh |
| 12 | 80h |
| 13 | 8Dh |
| 14 | 8Bh |
| 15 | 8Eh |

N_FTS defines the number of FTSs that must be transmitted when transitioning from L0s to L0. At the 2.5 GT/s data rate, the value that can be requested by a component corresponds to a Symbol lock time of 16 ns (N_FTS set to 0b and one SKP Ordered Set) to ~4 µs (N_FTS set to 255), except when the Extended Synch bit is Set, which requires the transmission of 4096 FTSs resulting in a bit lock time of 64 µs. For 8.0 GT/s and above data rates, when the Extended Synch bit is Set, the transmitter is required to send 4096 FTS Ordered Set Blocks. Note that the N_FTS value reported by a component may change; for example, due to software modifying the value in the Common Clock Configuration bit (see § Section 7.5.3.7).

If the N_FTS period of time expires before the Receiver obtains bit lock, Symbol lock or Block alignment, and Lane-to-Lane de-skew on all Lanes of the configured Link, the Receiver must transition to the Recovery state. This sequence is detailed in the LTSSM in § Section 4.2.6 .

4.2.5.7 Start of Data Stream Ordered Set (SDS Ordered Set) §

The Start of Data Stream (SDS) Ordered Set, described in § Table 4-52 , § Table 4-53 , and § Table 4-54 is defined only for 128b/130b encoding and 1b/1b encoding. It is transmitted in the Configuration.Idle , Recovery.Idle , and Tx_L0s.FTS LTSSM states to define the transition from Ordered Set Blocks to a Data Stream, and Loopback Leads are permitted to transmit it as described in § Section 4.2.2.6 . It must not be transmitted at any other time. While not in the Loopback state, the Block following an SDS Ordered Set must be a Data Block in Non-Flit Mode, and the first Symbol of that Data Block is the first Symbol of the Data Stream. In 1b/1b encoding, the Transmitter must send two back to back SDS when the conditions to send an SDS are met. The SDS Ordered Set in 1b/1b encoding must be on an aligned 128b (16B) boundary. An SDS Ordered Set sequence refers to either the single SDS Ordered set with 128b/130b encoding or the two back to back SDS Ordered Sets with 1b/1b encoding. In Flit Mode, with 128b/130b encoding and 1b/1b encoding, a Control SKP Ordered Set must be sent after the SDS Ordered Set . The first Symbol of the Data Stream starts immediately after the Control SKP Ordered Set . With 1b/1b encoding, a Receiver considers an SDS sequence valid if four good B1_C6_C6_C6 (4B) sets are received, at least two of which are in an even 4 byte aligned position (i.e., bytes 0-3, 8-11).

Table 4-52 SDS Ordered Set (for 8.0 GT/s and 16.0 GT/s Data Rate) §

| Symbol Number | Value | Description |
|---------------|-------|----------------------------|
| 0 | E1h | SDS Ordered Set Identifier |
| 1-15 | 55h | Body of SDS Ordered Set |

Table 4-53 SDS Ordered Set (for 32.0 GT/s) §

| Symbol Number | Value | Description |
|---------------|-------|----------------------------|
| 0 | E1h | SDS Ordered Set Identifier |
| 1-15 | 87h | Body of SDS Ordered Set |

Table 4-54 SDS Ordered Set (for 64.0 GT/s) §

| Symbol Number | Value | Description |
|-----------------------|-------|----------------------------|
| 0, 4, 8, 12 | B1h | SDS Ordered Set Identifier |
| 1-3, 5-7, 9-11, 13-15 | C6h | Body of SDS Ordered Set |

4.2.5.8 Link Error Recovery §

- Link Errors, when operating with 8b/10b encoding are:
 - 8b/10b decode errors, Non-Flit Mode framing related errors defined in § Section 4.2.1.2.1 , loss of Symbol lock, Elasticity Buffer Overflow/Underflow, or loss of Lane-to-Lane de-skew.
 - 8b/10b decode errors must be checked and must trigger a Receiver Error in specified LTSSM states (see § Table 4-60), which is a reported error associated with the Port (see § Section 6.2). Triggering a Receiver Error on any or all of Non-Flit Mode framing related errors defined in § Section 4.2.1.2.1 , loss of Symbol Lock, Lane De-skew Error, and Elasticity Buffer Overflow/Underflow is optional.
- Link Errors, when operating with 128b/130b encoding, are:

- Framing Errors, loss of Block Alignment, Elasticity Buffer Overflow/Underflow, or loss of Lane-to-Lane de-skew.
- Framing errors must be checked and trigger a Receiver Error in the LTSSM states specified in § Table 4-60 . The Receiver Error is a reported error associated with the Port (see § Section 6.2). Triggering a Receiver Error on any of all of loss of Block Alignment, Elasticity Buffer Overflow/Underflow, and loss of Lane-to-Lane de-skew is optional.
- Link Errors, when operating with 1b/1b encoding, are:
 - Framing Errors, Elasticity Buffer Overflow/Underflow, or loss of Lane-to-Lane de-skew.
 - Framing errors must be checked and trigger a Receiver Error in the LTSSM states specified in § Table 4-60 . The Receiver Error is a reported error associated with the Port (see § Section 6.2). Triggering a Receiver Error on any of all of Elasticity Buffer Overflow/Underflow, and loss of Lane-to-Lane de-skew is optional.
- On a configured Link, which is in L0, error recovery will at a minimum be managed in a Layer above the Physical Layer (as described in § Section 3.6) by directing the Link to transition to Recovery.
 - Note: Link Errors may also result in the Physical Layer initiating an LTSSM state transition from L0 to Recovery.
- All LTSSM states other than L0 make progress⁸² when Link Errors occur.
 - When operating with 8b/10b encoding, Link Errors that occur in LTSSM states other than L0 must not result in the Physical Layer initiating an LTSSM state transition.
 - When operating with 128b/130b encoding and not processing a Data Stream, Link Errors that occur in LTSSM states other than L0 must not result in the Physical Layer initiating an LTSSM state transition.
- When operating with 8b/10b encoding, if a Lane detects an implementation specific number of 8b/10b errors, Symbol lock must be verified or re-established as soon as possible.⁸³

4.2.5.9 Reset §

Reset is described from a system point of view in § Section 6.6 .

4.2.5.9.1 Fundamental Reset §

When Fundamental Reset is asserted:

- The Receiver terminations are required to meet $Z_{RX-HIGH-IMP-DC-POS}$ and $Z_{RX-HIGH-IMP-DC-NEG}$ (see § Table 8-12).
- The Transmitter is required only to meet $I_{TX-SHORT}$ (see § Table 8-7).
- The Transmitter holds a constant DC common mode voltage.⁸⁴

When Fundamental Reset is deasserted:

- The Port LTSSM (see § Section 4.2.6) is initialized (see § Section 6.6.1 for additional requirements).

82. In this context, progress is defined as the LTSSM not remaining indefinitely in one state with the possible exception of Detect, or Disabled.

83. The method to verify and re-establish Symbol lock is implementation specific.

84. The common mode being driven is not required to meet the Absolute Delta Between DC Common Mode during L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).

4.2.5.9.2 Hot Reset §

Hot Reset is a protocol reset defined in § [Section 4.2.6.12](#).

4.2.5.10 Link Data Rate Negotiation §

All devices are required to start Link initialization using a 2.5 GT/s data rate on each Lane. A field in the training sequence Ordered Set (see § [Section 4.2.5.1](#)) is used to advertise all supported data rates. The Link trains to L0 initially in 2.5 GT/s data rate after which a data rate change occurs by going through the Recovery state.

4.2.5.11 Link Width and Lane Sequence Negotiation §

PCI Express Links must consist of 1, 2, 4, 8, or 16 Lanes in parallel, referred to as x1, x2, x4, x8, and x16 Links, respectively. All Lanes within a Link must simultaneously transmit data based on the same frequency with a skew between Lanes not to exceed L_{TX-SKEW} (§ [Table 8-6](#)). The negotiation process is described as a sequence of steps.

The negotiation establishes values for Link number and Lane number for each Lane that is part of a valid Link; each Lane that is not part of a valid Link exits the negotiation to become a separate Link or remains in Electrical Idle.

During Link width and Lane number negotiation, the two communicating Ports must accommodate the maximum allowed Lane-to-Lane skew as specified by L_{RX-SKEW} in § [Table 8-12](#).

Optional Link negotiation behaviors include Lane reversal, variable width Links, splitting of Ports into multiple Links and the configuration of a crosslink.

Other specifications may impose other rules and restrictions that must be comprehended by components compliant to those other specifications; it is the intent of this specification to comprehend interoperability for a broad range of component capabilities.

4.2.5.11.1 Required and Optional Port Behavior §

- The ability for a xN Port to form a xN Link as well as a x1 Link (where N can be 16, 8, 4, 2, and 1) is required.
 - Designers must connect Ports between two different components in a way that allows those components to meet the above requirement. If the Ports between components are connected in ways that are not consistent with intended usage as defined by the component's Port descriptions/ data sheets, behavior is undefined.
- The ability for a xN Port to form any Link width between N and 1 is optional.
 - An example of this behavior includes a x16 Port which can only configure into only one Link, but the width of the Link can be configured to be x8, x4, x2 as well as the required widths of x16 and x1.
- The ability to split a Port into two or more Links is optional.
 - An example of this behavior would be a x16 Port that may be able to configure two x8 Links, four x4 Links, or 16 x1 Links.
- Support for Lane reversal is optional.
 - If implemented, Lane reversal must be done for both the Transmitter and Receiver of a given Port for a multi-Lane Link.

- An example of Lane reversal consists of Lane 0 of an Upstream Port attached to Lane N-1 of a Downstream Port where either the Downstream or Upstream device may reverse the Lane order to configure a xN Link.

Support for formation of a crosslink is optional. In this context, a Downstream Port connected to a Downstream Port or an Upstream Port connected to an Upstream Port is a crosslink.

Current and future electromechanical and/or form factor specifications may require the implementation of some optional features listed above. Component designers must read the specifications for the systems that the component(s) they are designing will used in to ensure compliance to those specifications.

4.2.5.12 Lane-to-Lane De-skew §

The Receiver must compensate for the allowable skew between all Lanes within a multi-Lane Link (see § Table 8-7 and § Table 8-12) before delivering the data and control to the Data Link Layer.

When using 8b/10b encoding, an unambiguous Lane-to-Lane de-skew mechanism may use one or more of the following:

- The COM Symbol of a received TS1 or TS2 Ordered Set
- The COM Symbol of a received Electrical Idle Exit Ordered Set
- The COM Symbol of the first received SKP Ordered Set after an FTS sequence
- The COM Symbol of a received SKP Ordered Set during a training sequence when not using SRIS.

When using 128b/130b encoding, an unambiguous Lane-to-Lane de-skew mechanism may use one or more of the following:

- A received SDS Ordered Set
- A received Electrical Idle Exit Ordered Set except when exiting L0s
- The first received Electrical Idle Exit Ordered Set after an FTS Ordered Set when exiting L0s
- When operating at 8.0 GT/s, a received SKP Ordered Set
- When operating at a data rate of 16.0 GT/s or higher, the first received SKP Ordered Set after an FTS sequence
- When operating at a data rate of 16.0 GT/s or higher, a received SKP Ordered Set except when:
 - exiting a training sequence or
 - two SKP Ordered Sets are separated by an EDS

When using 1b/1b encoding, an unambiguous Lane-to-Lane de-skew mechanism may use one or more of the following:

- A received SDS Ordered Set
- A received Electrical Idle Exit Ordered Set
- A received Control SKP Ordered Set

Other de-skew mechanisms may also be employed, provided they are unambiguous. Lane-to-Lane de-skew must be performed during Configuration, Recovery, and L0s in the LTSSM.

IMPLEMENTATION NOTE: UNAMBIGUOUS LANE-TO-LANE DE-SKEW: §

The max skew at 2.5 GT/s that a Receiver must be able to de-skew is 20 ns. A nominal SKP Ordered Set (i.e., one that does not have SKP Symbols added or removed by a Retimer) is 4 Symbols long, or 16 ns, at 2.5 GT/s.

~~↑↑Generally, ↑↑Generally, ↑~~ SKP Ordered Sets are transmitted such that they are well spaced out, and no particular care is needed to use them for de-skew (i.e., they provide an unambiguous mechanism). If back-to-back SKP Ordered Sets are transmitted, an implementation that simply looks for the COM of the SKP Ordered Set to occur on each Lane at the same point in time may fail. When exiting L0s a transmitter may send back-to-back SKP Ordered Sets after the last FTS Ordered Set of the Fast Training Sequence. De-skew must be obtained in L0s, therefore the implementation must comprehend back-to-back SKP Ordered Sets when performing de-skew in this case.

Exceptions to the unambiguous mechanism in § Section 4.2.5.12 occur because back-to-back Ordered Sets might be sent (i.e., EIEOS might be sent back-to-back when exiting L0s when using 128b/130b encoding.) EIEOS can still be used for de-skew in this case, however the implementation must comprehend back-to-back EIEOS when performing de-skew.

When operating at a data rate of 16.0 GT/s or higher, a transmitter may send back-to-back SKP Ordered Sets at the end of a Training Sequence (e.g., TS2 Ordered Set, SKP Ordered Set, SKP Ordered Set, SDS Ordered Set). Implementations that choose to use SKP Ordered Sets for de-skew in this case are recommended to recognize that the back-to-back SKP Ordered Sets are different (i.e., Standard SKP Ordered Set followed by Control SKP Ordered Set).

4.2.5.13 Lane vs. Link Training §

The Link initialization process builds unassociated Lanes of a Port into associated Lanes that form a Link. For Lanes to configure properly into a desired Link, the TS1 and TS2 Ordered Sets must have the appropriate fields (Symbol 3, 4, and 5) set to the same values on all Lanes.

Links are formed at the conclusion of Configuration.

- If the optional behavior of a Port being able to configure multiple Links is employed, the following observations can be made:
 - A separate LTSSM is needed for each separate Link that is desired to be configured by any given Port.
 - The LTSSM Rules are written for configuring one Link. The decision to configure Links in a serial fashion or parallel is implementation specific.

4.2.6 Link Training and Status State Machine (LTSSM) Descriptions §

The LTSSM states are illustrated in § Figure 4-67. These states are described in following sections.

All timeout values specified for the Link Training and Status state machine (LTSSM) are minus 0 seconds and plus 50% unless explicitly stated otherwise. All timeout values must be set to the specified values after Fundamental Reset. All counter values must be set to the specified values after Fundamental Reset.

4.2.6.1 Detect Overview §

The purpose of this state is to detect when a far end termination is present.

4.2.6.2 Polling Overview §

The Port transmits training Ordered Sets and responds to the received training Ordered Sets. In this state, bit lock and Symbol lock are established and Lane polarity is configured.

The Polling state includes Polling.Compliance (see § Section 4.2.7.2.2). This state is intended for use with test equipment used to assess if the Transmitter and the interconnect present in the device under test setup is compliant with the voltage and timing specifications in § Table 8-6 , § Table 8-7 , and § Table 8-12 .

The Polling.Compliance state also includes a simplified ~~↓↓interoperability↓↓interoperability~~ testing scheme that is intended to be performed using a wide array of test and measurement equipment (i.e., pattern generator, oscilloscope, BERT, etc.). This portion of the Polling.Compliance state is logically entered by at least one component transmitting a TS1 with Compliance_Receive_Request asserted upon entering Polling.Active . The ability to assert Compliance_Receive_Request is implementation specific. A provision for changing data rates to that indicated by the highest common transmitted and received Data Rate Identifiers (Symbol 4 of TS1) is also included to make this behavior scalable to various data rates.

IMPLEMENTATION NOTE: USE OF POLLING.COMPLIANCE §

Polling.Compliance is intended for a compliance test environment and not entered during normal operation and cannot be disabled for any reason. Polling.Compliance is entered based on the physical system environment or configuration register access mechanism as described in § Section 4.2.7.2.1 . Any other mechanism that causes a Transmitter to output the compliance pattern is implementation specific and is beyond the scope of this specification.

4.2.6.3 Configuration Overview §

In Configuration , both the Transmitter and Receiver are sending and receiving data at the negotiated data rate. The Lanes of a Port configure into a Link through a width and Lane negotiation sequence. Also, Lane-to-Lane de-skew must occur, scrambling can be disabled if permitted, the N_FTS is set, and the Disabled or Loopback states can be entered.

4.2.6.4 Recovery Overview §

In Recovery , both the Transmitter and Receiver are sending and receiving data using the configured Link and Lane number as well as the previously supported data rate(s). Recovery allows a configured Link to change the data rate of operation if desired, re-establish bit lock, Symbol lock or Block alignment, and Lane-to-Lane de-skew. Recovery is also used to set a new N_FTS value and enter the Loopback, Disabled , Hot Reset , and Configuration states.

4.2.6.5 L0 Overview §

L0 is the normal operational state where data and control packets can be transmitted and received. All power management states are entered from this state.

4.2.6.6 L0s Overview §

L0s is intended as a power savings state.

- L0s is not supported in Flit Mode and hardware must ignore the value of the L0s Enable bit for a Link operating in FM,
- L0s is not supported on a Link that contains Retimers, or
- L0s is not supported when operating with separate reference clocks with independent Spread Spectrum Clocking (SSC) (see § Section 4.2.8 and § Section 4.3.10).

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

The entry to L0s occurs after receiving an EIOS .

The exit from L0s to L0 must re-establish bit lock, Symbol lock or Block alignment, and Lane-to-Lane de-skew.

A Transmitter and Receiver Lane pair on a Port are not required to both be in L0s simultaneously.

4.2.6.7 L0p Overview §

L0p is a part of the L0 state and intended as a power savings state. L0p is supported only in Flit Mode, for all data rates. L0p support is optional but strongly recommended for Ports. L0p support is mandatory for Pseudo-Ports (Retimers) that support Flit Mode. L0p enables a Link to have some Lanes active while the remaining Lanes will be in electrical idle state. With Flit Mode, Link Upconfigure, which performs dynamic link width adjustment through the state transition L0 → Recovery → Configuration → L0 , is not supported.

L0p is symmetric in width. All legal widths (x1, x2, x4, x8, x16) up to the configured Link width must be supported by Ports that support L0p . When the Link is in L0p , at least one Lane in each direction must be active. L0p support is negotiated only during Configuration.Complete state when LinkUp=0b, as described in § Section 4.2.7 .

Device  Port Functions indicate support for L0p by Setting L0p Supported and by supplying a Port L0p Exit Latency value.

Errata: Base 6.3
B823△◀▷



 Port Function for which L0p Enable is set to 1b, Hardware Autonomous Width Disable is set to 0b, and Target Link Width is set to 111b, is permitted to initiate L0p requests with no architected software intervention by sending Link Management DLLPs (see § Section 4.2.6.7.1) to its Link Partner. System software can direct   Port Function to initiate an L0p Link Width change by sending to that Function a CfgWr TLP to set L0p Enable to 1b and Target Link Width to a value from 000b through 100b. It is strongly recommended that system software leave Target Link Width unchanged (i.e., 111b) other than for usage cases such as test or debug.

Errata: Base 6.3
B823△◀▷

If the link is L0p capable and the Link width has been changed by entering the Configuration state, the following rules apply:

1. On entry to Recovery the Link will revert to its configured Link width on the last entry to L0

2. L0p cannot be used to turn on Lanes that were turned off in Configuration state.

IMPLEMENTATION NOTE: AVAILABILITY OF LANES TURNED OFF IN CONFIGURATION §

The Lanes that are turned off during Configuration state may be due to issues such as reliability and are no longer associated with the LTSSM while LinkUp=1b (see § Section 4.2.7.3).

Errata: Base 6.3
B827△◀

↑↑A Port is strongly recommended to not use link-width downsize while operating at Data Rates lower than 16.0 GT/s if Retimers are present, unless it knows using implementation specific mechanism that the Retimers in use can pass the Ordered Sets prior to obtaining lane-to-lane deskew.↑

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: USE OF L0P WITH MULTIPLE RETIMERS AT DATA RATES BELOW 16.0 GT/S §

Errata: Base 6.3
B827△◀

↑↑With multiple Retimers at data rates below 16.0 GT/s, the link width upsize may not be able to meet the 16 µs timeout limit after a Port Acknowledges the up-size request, as illustrated by the following example of two Retimers connecting a Root Port (RP) to an Endpoint (EP), operating at 2.5 GT/s.↑

1. ↑↑RP initiates an L0p up-size request. With 2 Retimers in the system, EP receives these requests after 2 Retimer latency.↑
2. ↑↑EP sends an Ack response and starts the 16 µs timer as per § Section 4.2.6.7 for it to expect EIEOS / TS1 on additional lanes.↑
3. ↑↑RP receives the Ack response after 2 Retimer latency and then takes approximately 6 µs (1538 symbol times) to start sending SKP OS.↑
4. ↑↑1st Retimer takes an additional 6 µs to align the SKP OS on the newly added lanes with existing lanes.↑
5. ↑↑2nd Retimer takes an additional 6 µs to align the SKP OS on the newly added lanes with existing lanes. (Total 18 µs)↑
6. ↑↑Meanwhile EP times out after 16 µs and considers the L0p up-size request to be abandoned.↑
7. ↑↑RP times out after 24 ms and goes to Recovery .↑

With L0p , any Lane that goes to electrical idle must be in electrical idle for a minimum time of $T_{TX-IDLE-MIN}$. The DC common mode voltage in TX during electrical idle must be within specification. The Receiver needs to wait a minimum of $T_{TX-IDLE-MIN}$ to start looking for Electrical Idle Exit.

Once L0p Is enabled, any Port can request a Link width change by sending up to three identical Link Management DLLPs within five consecutive Flits. Two Flits are considered consecutive even if they are separated by an Ordered Set. This can be either up-size to increase the number of active Lanes or down-size to reduce the number of active Lanes. If L0p Enable is Set, the Link partner must either Ack or Nak each L0p request on a valid Flit , unless it is permitted to ignore the request. If L0p Enable is Clear or Hardware Autonomous Width Disable is Set, the Link partner is strongly recommended to Nak the L0p request unless the request is an upsize to the fully configured link width. A Port must respond to an L0p request within the following time interval of receiving a valid Flit with the request: 4 µs if 8b/10b encoding is used, 1 µs

otherwise. Time is measured from the last bit of the Flit with the request on the ingress package pin to the first bit of the Flit with the response on the egress package pin. A Port that has requested a Link width change but has not received a response within the following time interval of issuing the request must either re-request the identical Link width change or abandon the request: 8 µs if 8b/10b encoding is used, 4 µs otherwise. If identical L0p requests are received within five consecutive Flits, the same response must be sent. After responding to a request, the Link partner must consider the request to be **abandoned** if one of the following conditions are true:

- On an up-size request:
 - the requester did not initiate the Link width upsize within 16 µs after the Flit with the Ack was transmitted AND
 - the requester did not re-send the up-size request within 8 µs after the Flit with the Ack was transmitted
- On a down-size request with an Ack response:
 - the requester did not resend the Link width downsize request within 4 µs after the Flit with the Ack was transmitted, AND
 - the requester did not initiate the Link width downsize within 16 µs after the Flit with the Ack was transmitted and the Link Partner did not down-size the Link by sending an EIOSQ
- On a down-size request with a Nak response:
 - the requester did not resend the Link width down-size request within 4 µs after the Flit with the Nak was transmitted

The following rules must be followed for L0p :

- If the Hardware Autonomous Width Disable bit in the Link Control register is set to 1b or if the L0p Enable bit is set to 0b, a Port:
 - Must not request any down-size.
 - Must initiate an up-size request to fully configured Link width if the Link is not at the fully configured width.
- L0p Ack and Nak Rules:
 - A Port is permitted to Nak an L0p Request for a lower width than the current width if L0p.Priority is not set.
 - If a Port is requesting link width down-size due to thermal throttling or reliability reasons, then it should set L0p.Priority in the L0p Request. The Link partner must Ack priority L0p Requests if it has L0p.Enable Set and Hardware Autonomous Width Disable is Clear, and if it is not requesting an L0p width change and it is not going to request an L0p width change in the next Flit (in which case, the rules for simultaneous link width change requests below must be followed). If the Link partner is going to request an L0p width change in the next Flit, it must show up on the Transmitter pins in 100 ns with L0p.Priority set in the L0p Request. The requirements for setting L0p.Priority are implementation specific and behavior is undefined if L0p.Priority bit is set on an upsize.
 - Any Port requesting a link width down-size that receives an Ack from its Link partner is responsible for initiating the up-size request eventually when the underlying conditions no longer prohibit the link from operating at full width.
 - If both Ports are simultaneously requesting link width change the following rules apply to determine which request wins. A Port is permitted to consider a pending request for L0p as simultaneous as long as it has the L0p request scheduled to appear in its Transmitter pins within 100 ns. The Port that does not win must Ack the request to the winning Port through the proper L0p response. The port that does win must either Nak the request from the losing Port or ignore the request.
 - If both requests have L0p.Priority set:
 - if both sides are requesting the same width, the Downstream Port wins

- else, the request with the lower width wins
- If one request has L0p.Priority set, it wins
- If neither side has L0p.Priority set:
 - if both sides are requesting the same width, the Downstream Port wins
 - else, the request with the higher width wins
- A Port must ignore another L0p Request with the same width if it has already Ack'ed an L0p request and its Transmitter Lanes are already at or in the process of transitioning to the same width.
- An entry to Recovery results in the Link going to its configured Link width on the most recent transition from Configuration to L0 and all the associated L0p transition states (such as width change request/response and all the tracking information such as time since Ack/ Nak) are reset as if no width change request was made.

Once an L0p request to change link width has been initiated, it must be completed or abandoned.

A Port must not initiate a request for a new Link width unless the following conditions are met:

1. No link width resizing has been in progress in the last 1 µs if 1b/1b or 128b/130b encoding is used or 4 µs if 8b/10b encoding is used.
2. If the Port abandoned its last L0p Request, at least 16 µs has elapsed since the request was abandoned.
3. If the Port Ack'ed the last L0p request from the remote Port, that request was either completed or abandoned.

On a Link width down-size, the following steps must be taken by each Port independently after the Link width down-configure request has been 'Ack'd:

- On the next scheduled SKP Ordered Set interval, the Lanes that are being turned off will send an EIOSQ instead of a SKP Ordered Set.
- The Lanes that sent an EIOSQ go to Electrical Idle after the EIOSQ is sent. The Lanes that sent a SKP Ordered Set resume transmitting Flits with a SKP Ordered Set insertion interval that corresponds to the new Link width (see § Section 4.2.3.4.6).
- If the requesting Port did not receive an Ack for the L0p request, but sees the Link Partner sending the EIOSQ Ordered Set, it must treat that as an 'Ack'. (An example scenario where this may happen is if the Flit containing the Ack DLLP was corrupted.)
- During the down-size negotiation, an EIOS must be received on all lanes to be deactivated at the same time after adjusting for lane to lane skew; else the LTSSM must enter Recovery.
 - A Port is permitted to enter Recovery after 24 msec of sending the EIOSQ on a downsize if the Link did not achieve the desired Link width in both directions

On a Link width up-size the following steps must be taken:

- The up-sizing action must be initiated by the requesting port. The non-requesting Port waits for detection of Electrical Idle exit on the lanes to be activated before starting the up-sizing actions.
- Data Stream continues on the active Lanes
- For the Lanes to be activated the following sequence must be followed:⁸⁵
 - SKP Ordered Sets must be scheduled at the same time as Lane 0 and the same number of SKPs must be added or deleted as Lane 0 by the Receiver in the (Pseudo-)Port. A Port is permitted to truncate a TS1/TS2 Ordered Set in 8b/10b encoding to transmit the SKP Ordered Set at the same time as Lane 0.

⁸⁵. This is similar but not identical to Recovery.

- Transmit TS1 Ordered Sets meeting the requirements for TS1 Ordered Sets in § Section 4.2.7.4.1 and § Section 4.2.5.3 . The Transmitter must ensure it follows the EIEOSQ rule of not being broken up by a SKP Ordered Set even though it has to send SKP Ordered Set at the same time as the active Lanes. This can be done by scheduling the start of EIEOSQ appropriately. However, a Transmitter is permitted to restart the EIEOSQ sequence after the SKP Ordered Set if the SKP Ordered Set interrupted an EIEOSQ in progress. If an EIEOSQ is scheduled to precede a subsequent scheduled SDS Ordered Set within 256 UI, the transmitter is permitted to not send that EIEOSQ or to delay the SDS until the next SKP interval.
- If eight consecutive TS1 or TS2 Ordered sets are received on all Lanes that are to be activated, the transmitter must transition to sending TS2 Ordered Sets meeting the requirements for TS2 Ordered Sets in § Section 4.2.7.4.4 and § Section 4.2.5.3 . If the Extended Synch bit is Set, it is strongly recommended that the Transmitter send a minimum of 1024 consecutive TS1 Ordered Sets before transitioning to sending TS2 Ordered Sets.
- After receiving eight consecutive TS2 Ordered Sets and sending at least 16 TS2 Ordered sets after the receipt of one TS2 Ordered Set on all the Lanes to be activated, the Port sends an SDS Ordered Set sequence if the data rate is 8.0 GT/s or higher just prior to sending the next scheduled SKP Ordered Set that will be sent (on the active as well as to be activated Lanes).
- Lane to Lane de-skew must be completed by the Receiver using the SKP Ordered Set across all the currently active as well as the Lanes being activated that have received eight consecutive TS2 Ordered Set and the SDS Ordered Set, if the data rate is 8.0 GT/s or higher. Starting with the SKP Ordered Set that follows the SDS Ordered Set, the SKP Ordered Set insertion interval that corresponds to the new Link width takes effect (see § Section 4.2.3.4.6).
- If 24 ms has elapsed since the start of activation and the Lanes are not part of the active Lanes, the LTSSM must be directed to enter Recovery .

The DLLP encoding for the various L0p commands and responses is shown in § Figure 3-15 . The definition of the various fields in the DLLP are shown in § Table 4-56 .

IMPLEMENTATION NOTE: : POPULATION OF TS1/TS2 ORDERED SET FIELDS DURING L0P §

When using 1b/1b Encoding, values to use for the various TS1/TS2 bits and fields are specified in § Table 4-39 . In many cases those values depend on the current LTSSM state, which is L0 during L0p Link width up-size. However, since the rules in this section indicate that the transmitted TS1 Ordered Sets should meet the requirements specified in § Section 4.2.7.4.1 , and that the transmitted TS2 Ordered Sets should meet the requirements specified in § Section 4.2.7.4.4 , implementations that populate those bits and fields as if the LTSSM state was Recovery.Rcvrlock for TS1 and Recovery.RcvrCfg for TS2 will not impede forward progress and are acceptable.

IMPLEMENTATION NOTE: ORTHOGONALITY OF L0P AND L1/L2 §

L0p is a reduced power sub-state of L0 . It is orthogonal to L1 and L2 . For a Port that is in the process of initiating or accepting an L1 or L2 request, it is recommended to not initiate, renew, or respond to an L0p request. Nevertheless, negotiations of L0p and L1 or L0p and L2 are permitted to occur concurrently and will ultimately resolve in one of the power states being negotiated, depending on the sequence of the negotiations and the desired intention of each Port. Refer to § Section 5.3.2 for rules that govern Link power management.

IMPLEMENTATION NOTE: SUMMARY OF L0P TRANSMITTER/RECEIVER BEHAVIOR §

When Remote L0p Supported is set, the table below summarizes the L0p Transmitter and Receiver behavior discussed above. The table below applies independently to each end of the link. It is recommended that SW sets the Hardware Autonomous Width Disable and L0p Enable bits consistently on both sides of the link.

It is recommended that the Link is at full width prior to changing the L0p Enable setting.

It is recommended that an implementation specific repeated request limit should also be deployed.

Table 4-55 Summary of L0p Transmitter/Receiver Behavior §

| SW Control bits @ Tx/Rx | | Tx side behavior | | | Rx side behavior | | |
|-----------------------------------|------------|--|---|---|---|--|---|
| Hardware Autonomous Width Disable | L0p Enable | Issue Nonpriority L0p request for width reduction? | Issue Priority L0p request for width reduction? | Issue L0p request for width increase? | Response to Nonpriority L0p request for width reduction? | Response to Priority L0p request for width reduction? | Response to L0p request for width increase? |
| 1 | x | Not permitted | Not permitted | Must issue to achieve fully configured width, if at lower width | Recommend Nak. But 'No-response' is permitted as well. | Recommend Nak. But 'No-response' is permitted as well. | Ack for full width increase. Otherwise, recommend Nak but 'No-response' is permitted as well. |
| 0 | 0 | Not permitted | Not permitted | Must issue to achieve fully configured width, if at lower width | Nak | Nak | Ack for full width increase. Otherwise, recommend Nak but 'No-response' is permitted as well. |
| 0 | 1 | Permitted | Permitted | Permitted | Ack/Nak <small>(with exception noted in § Section 4.2.6.7)</small> | | Ack |

4.2.6.7.1 Link Management DLLP §

Table 4-56 Link Management DLLP §

| Field | Description | | |
|--------|---|--|--|
| Byte 0 | Link Management DLLP - must be 0010 1000b | | |
| Byte 1 | Link Mgmt Type - qualifies Bytes 2 and 3 0000 0000b L0p DLLP Others Reserved | | |

Base 6.4 vs Base 6.3

| Field | Description | |
|--|--|---|
| <i>L0p.Priority</i> - L0p Priority Request - Meaning of this field varies by <u>L0p.Cmd</u> value | | |
| Byte 2, Bit [4] | If <u>L0p.Cmd</u> is: | Description is: |
| | <u>L0p Request</u> | Request Priority 0b Normal Priority L0p Request 1b High Priority L0p Request |
| | <u>Others</u> | Reserved |
| Reserved if <u>Link Mgmt Type</u> is other than 00h. | | |
| Byte 2, Bits [3:0] | <i>L0p.Cmd</i> - L0p Command or Response | |
| | 0100b | <i>L0p Request</i> |
| | 0110b | <i>L0p Request Ack</i> |
| Byte 3, Bits [7:4] | 0111b | <i>L0p Request Nak</i> |
| | Others | Reserved |
| | Reserved if <u>Link Mgmt Type</u> is other than 00h. | |
| Meaning of this field varies by <u>L0p.Cmd</u> value | | |
| Byte 3, Bits [7:4] | If <u>L0p.Cmd</u> is: | Description is: |
| | <u>L0p Request Ack</u> | <i>Response Payload</i> - Reflects the value of <u>L0p Link Width</u> of the corresponding command 0001b x1 0010b x2 0100b x4 1000b x8 0000b x16 |
| | <u>L0p Request Nak</u> | Others Reserved |
| Others | | |
| Reserved if <u>Link Mgmt Type</u> is other than 00h. | | |
| Meaning of this field varies by <u>L0p.Cmd</u> value | | |
| Byte 3, Bits [3:0] | If <u>L0p.Cmd</u> is: | Description is: |
| | <u>L0p Request</u> | <i>L0p Link Width</i> - Desired Link Width 0001b x1 0010b x2 0100b x4 1000b x8 0000b x16 |
| | <u>Others</u> | Reserved |

| Field | Description | |
|---|-----------------|----------|
| If L0p.Cmd is: | Description is: | |
| | Others | Reserved |
| | Others | Reserved |
| Reserved if Link Mgmt Type is other than 00h. | | |

Receivers must silently ignore Link Management DLLPs if:

- The Link is operating in Non-Flit Mode.
- The Link Mgmt Type field contains a Reserved value.
- The Link Mgmt Type field is 00h and the L0p.Cmd field contains a reserved value.
- The Link Mgmt Type field is 00h, the L0p.Cmd field contains L0p Request and the L0p Link Width field contains a reserved value.
- The Link Mgmt Type field is 00h, the L0p.Cmd field contains L0p Request Ack or L0p Request Nak and Response Payload contains a reserved value.

IMPLEMENTATION NOTE: L0P ENTRY / EXIT TIMES §

The Lanes that are electrically idle are expected to have power savings and entry/ exit times similar to L1. The deeper the power savings, the higher the exit latency to bring those idle Lanes back to active state. For example, a PLL associated exclusively with the set of Lanes that are idle can be turned off. This will result in better power savings but the exit time to activate the Lanes will be higher.

Receiver hardware should use the L0p Exit Latency values from the Data Link Feature DLLP to help determine when to request L0p. These values are visible to software in the Local L0p Exit Latency and Remote L0p Exit Latency fields.

System software should ensure that the exit latency of all Retimers are included some Retimer L0p Exit Latency field (either Upstream or Downstream Port). In general, Retimers located on an add-in card should be included in the add-in card's Upstream Port's Retimer L0p Exit Latency and Retimers located in the system chassis should be included in the Downstream Port's Retimer L0p Exit Latency .

Example of L0p Flows: § Figure 4-66 demonstrates L0p flows in a x16 Link.

1. L0p is enabled on the Link during the Configuration.Complete state.
2. The USP requests the Link to downsize to x8 which is Ack'ed by the DSP.
3. On the next SKP Ordered Set the L0p indication is provided by the Port on a per-Lane basis. Lanes 0-7 continue with the traffic whereas Lanes 8-15 send an EIOSQ and go to electrical Idle. Thus, the Link now has 8 active Lanes and 8 Lanes in idle state.
4. At a later point, the DSP requests upsizing the Link to x16 while the USP has made a request to further downsize to x4. The x4 request gets NAK'ed, the x16 gets Ack'ed. Then link training proceeds in Lanes 8-15, initiated by the DSP with the exchange of TS1 / TS2 Ordered Sets with EIEOS inserted at appropriate intervals.

When Lanes 8-15 are ready, an SDS is sent immediately preceding the next scheduled SKP Ordered Set in the Lanes.

- The Link operates as a x16 Link after sending the SKP Ordered Set across all 16 Lanes. Even though L0p is symmetric during the transitions, the two sides may be operating at different widths for a while. This is similar to the situation during entry to L1 or L0.

Note that if the data rates is 2.5 GT/s, the EIEOS and SDS will not be present. If the data rate is 5.0 GT/s, EIEOS will be present but not SDS.

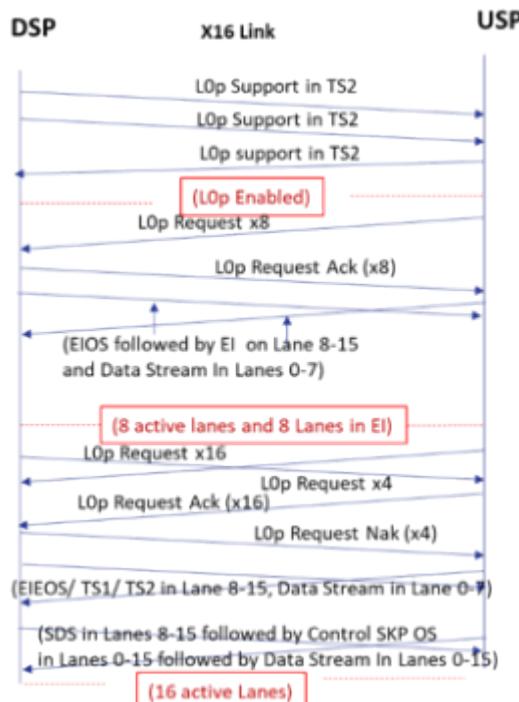


Figure 4-66 Example of L0p flow in a x16 Link §

4.2.6.8 L1 Overview §

L1 is intended as a power savings state.

The L1 state allows an additional power savings over L0s at the cost of additional resume latency.

The entry to L1 occurs after being directed by the Data Link Layer and receiving an EIOS.

4.2.6.9 L2 Overview §

Power can be aggressively conserved in L2. Most of the Transmitter and Receiver may be shut off.⁸⁶ Main power and clocks are not guaranteed, but Aux⁸⁷ power is available.

When Beacon support is required by the associated system or form factor specification, an Upstream Port that supports the wakeup capability must be able to send; and a Downstream Port must be able to receive; a wakeup signal referred to as a Beacon.

86. The exception is the Receiver termination, which must remain in a low impedance state.

87. In this context, Aux power means a power source which can be used to drive the Beacon circuitry.

The entry to L2 occurs after being directed by the Data Link Layer and receiving an EOS .

4.2.6.10 Disabled Overview §

The intent of the Disabled state is to allow a configured Link to be disabled as long as directed or until Electrical Idle is exited (i.e., due to a hot removal and insertion) after entering Disabled .

4.2.6.11 Loopback Overview §

Loopback is intended for test and fault isolation use. Only the entry and exit behavior is specified, all other details are implementation specific. Loopback can operate on either a per-Lane or configured Link basis.

A **Loopback Lead** is the component requesting Loopback .

A **Loopback Follower** is the component looping back the data.

In 8b/10b and 128b/130b encoding, Loopback uses bit 2 (Loopback) in the Training Control Field of TS1 and TS2 Ordered Sets (see § Table 4-36 and § Table 4-37). In 1b/1b encoding, Loopback uses an encoding of bits 3:0 in the Training Control field of TS1 and TS2 Ordered Sets (see § Table 4-39).

The entry mechanism for a Loopback Lead is device specific.

The Loopback Follower device enters Loopback whenever two consecutive TS1 Ordered Sets are received with Loopback_Request asserted.

IMPLEMENTATION NOTE: USE OF LOOPBACK §

Once in the Loopback state, the Lead can send any pattern of Symbols as long as the encoding rules are followed. Once in Loopback , the concept of data Loopback Follower simply transmits back the bit stream it receives without any modification for additional scrambling nor precoding. Retimers between the Loopback Lead and the Loopback Follower are unaware that Loopback is no longer relevant; what is sent out is looped back. taking place and continue to apply scrambling and precoding as before Loopback operation was established. The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to enter the Loopback state is implementation specific and beyond the scope of this specification.

Errata: Base 6.3

B831△◀

IMPLEMENTATION NOTE: LOOPBACK LEAD AND LOOPBACK FOLLOWER REPLACING OLDER TERMS §

The terms Loopback Lead and Loopback Follower are newer terms while preserving the identical Loopback functionality as in [PCIe-5.0]. These terms appropriately represent the functionality performed by the component. Any Port or a test equipment can be a Loopback Lead if it has that optional capability. Every Port is required to be a Loopback Follower . The test set up needs to comprehend that in such a way that it designates one Port to be the Loopback Lead and its Link Partner to be the Loopback Follower .

4.2.6.12 Hot Reset Overview §

The intent of the Hot Reset state is to allow a configured Link and associated downstream device to be reset using in-band signaling.

4.2.7 Link Training and Status State Rules §

Various Link status bits are monitored through software with the exception of **LinkUp** which is monitored by the Data Link Layer. § Table 4-60 describes how the Link status bits must be handled throughout the LTSSM (for more information, see § Section 3.2 for **LinkUp**; § Section 7.5.3.8 for Link Speed, Link Width, and Link Training; § Section 6.2 for Receiver Error; and § Section 6.7 for In-Band Presence Detect). A Receiver may also optionally report an 8b/10b Error in the Lane Error Status Register when operating in 8b/10b encoding, when allowed to report the error as a Receiver Error in § Table 4-60.

IMPLEMENTATION NOTE: RECEIVER ERRORS DURING CONFIGURATION AND RECOVERY STATES §

Allowing Receiver Errors to be set while in Configuration or Recovery is intended to allow implementations to report Link Errors that occur while processing packets in those states. For example, if the LTSSM transitions from L0 to Recovery while a TLP is being received, a Link Error that occurs after the LTSSM transition can be reported.

Table 4-60 Link Status Mapped to the LTSSM §

| LTSSM State | Link Width | Link Speed | LinkUp | Link Training | Receiver Error | In-Band Presence Detect ⁸⁸ |
|---------------|------------|---|---------------------|---------------|--|---------------------------------------|
| Detect | Undefined | Undefined | 0b | 0b | No action | 0b |
| Polling | Undefined | Set to 2.5 GT/s on entry from Detect. Link speed may change on entry to Polling.Compliance . | 0b | 0b | No action | 1b |
| Configuration | Set | No action | 0b/1b ⁸⁹ | 1b | Set on 8b/10b Error. Optional: Set on Link Error when using 128b/130b encoding. | 1b |
| Recovery | No action | Set to new speed when speed changes | 1b | 1b | Optionally set on Link Error. | 1b |
| L0 | No action | No action | 1b | 0b | Set on Link Error. | 1b |
| L0s | No action | No action | 1b | 0b | No action | 1b |

88. In-band refers to the fact that no sideband signals are used to calculate the presence of a powered up device on the other end of a Link.

89. LinkUp will always be 0 if coming into Configuration via Detect → Polling → Configuration and LinkUp will always be 1 if coming into Configuration from any other state.

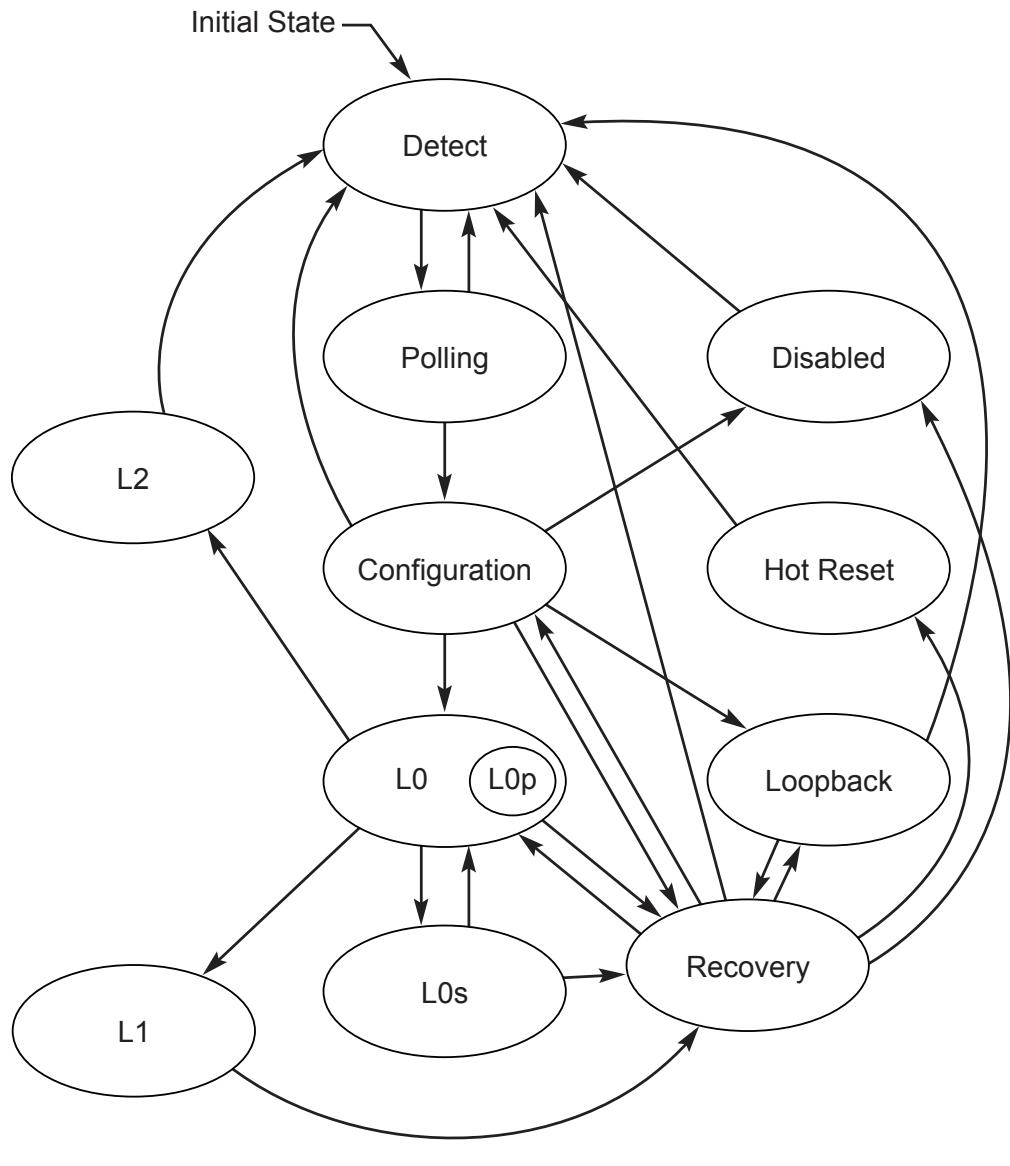
| LTSSM State | Link Width | Link Speed | LinkUp | Link Training | Receiver Error | In-Band Presence Detect |
|------------------|------------|---|--------|---------------|-------------------------------|-------------------------|
| <u>L1</u> | No action | No action | 1b | 0b | No action | 1b |
| <u>L2</u> | No action | No action | 1b | 0b | No action | 1b |
| <u>Disabled</u> | Undefined | Undefined | 0b | 0b | Optional: Set on 8b/10b Error | 1b |
| <u>Loopback</u> | No action | Link speed may change on entry to Loopback from Configuration . | 0b | 0b | No action | 1b |
| <u>Hot Reset</u> | No action | No action | 0b | 0b | Optional: Set on 8b/10b Error | 1b |

The state machine rules for configuring and operating a PCI Express Link are defined in the following sections.

Matching Link and Lane Numbers for 1b/1b Encoding

With 1b/1b encoding, the Link and Lane numbers are not sent explicitly in the TS1 / TS2 Ordered Sets in any States other than those listed in § Table 4-39 . Thus, when using 1b/1b encoding in any LTSSM state other than where Link and Lane numbers are transmitted, any references to an exact match in the Link and/or Lane numbers between the transmitted and received TS0 / TS1 / TS2 Ordered Sets must be assumed to be true as long as the Ordered Set is a valid 1b/1b TS0, TS1, or TS2 Ordered Set . In Recovery.RcvrCfg , if TS2 Ordered Sets with an Equalization Request (Equalization Byte 0: Bit 0=0b and Bit 7=1b) are received, this must be treated as an exact match in the Link number. If transmitting TS2 Ordered Sets with an Equalization Request and TS2 Ordered Sets with Link Number (or PAD) are received, Receivers must check the received Link number for a match with the Link number negotiated during the Configuration LTSSM state. Similarly, when using 1b/1b encoding in any LTSSM state other than Configuration , any references to transmitting a TS0 / TS1 / TS2 Ordered Set using the Link number and/or Lane number negotiated during the Configuration LTSSM state must be interpreted as using the Lane number for the purposes of scrambling in the TS0 / TS1 / TS2 Ordered Set.

Base 6.4 vs Base 6.3



OM13800F

Figure 4-67 Main State Diagram for Link Training and Status State Machine §

4.2.7.1 Detect §

The Detect substate machine is shown in § Figure 4-68 .

4.2.7.1.1 Detect.Quiet §

- Transmitter is in an Electrical Idle state.
 - The DC common mode voltage is not required to be within specification.

- 2.5 GT/s data rate is selected as the frequency of operation. If the frequency of operation was not 2.5 GT/s data rate on entry to this substate, the LTSSM must stay in this substate for at least 1 ms, during which the frequency of operation must be changed to the 2.5 GT/s data rate.
 - Note: This does not affect the advertised data rate in the TS1 and TS2 Ordered Sets.
- All Receivers must meet the Z_{RX-DC} specification for 2.5 GT/s within 1 ms (see § Table 8-12) of entering this substate. The LTSSM must stay in this substate until the Z_{RX-DC} specification for 2.5 GT/s is met.
- LinkUp = 0b (status is cleared).
- The Equalization 8.0 GT/s Phase 1 Successful, Equalization 8.0 GT/s Phase 2 Successful, Equalization 8.0 GT/s Phase 3 Successful, and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register are all set to 0b.

The Equalization 16.0 GT/s Phase 1 Successful, Equalization 16.0 GT/s Phase 2 Successful, Equalization 16.0 GT/s Phase 3 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register are all set to 0b.

The Equalization 32.0 GT/s Phase 1 Successful, Equalization 32.0 GT/s Phase 2 Successful, Equalization 32.0 GT/s Phase 3 Successful and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register are all set to 0b.

The Equalization 64.0 GT/s Phase 1 Successful, Equalization 64.0 GT/s Phase 2 Successful, Equalization 64.0 GT/s Phase 3 Successful, and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register are all set to 0b.

- The **use_modified_TS1_TS2_Ordered_Set** variable is reset to 0b.
- The Remote L0p Supported bit in the Device Status 3 Register is reset to 0b.
- The **Flit_Mode_Enabled** variable is reset to 0b. (When Flit_Mode_Enabled is set to 0b, the link will not operate in Flit Mode.)
- The **L0p_capable** variable is reset to 0b.
- The **SRIS_Mode_Enabled** variable is reset to 0b. When the SRIS_Mode_Enabled variable is set to 1b in Flit Mode, the SKP Ordered Set transmission frequency must follow the SRIS mode frequency for the data rate of operation. Otherwise, the SKP Ordered Set transmission frequency for the SRNS / Common Clock must be followed.
- The **directed_speed_change** variable is reset to 0b. The **upconfigure_capable** variable is reset to 0b. The **idle_to_rlock_transitioned** variable is reset to 00h. The **select_deemphasis** variable must be set to either 0b or 1b based on platform specific needs for an Upstream Port and identical to the Selectable Preset/De-emphasis bit in the Link Control 2 Register for a Downstream Port. The **equalization_done_8GT_data_rate**, **equalization_done_16GT_data_rate**, **equalization_done_32GT_data_rate**, and **equalization_done_64GT_data_rate** variables are reset to 0b. The **perform_equalization_for_loopback** and **perform_equalization_for_loopback_64GT** variables are reset to 0b.
 - Note that since these variables are defined with [PCIe-2.0], earlier devices would not implement these variables and will always take the path as if the directed_speed_change and upconfigure_capable variables are constantly reset to 0b and the idle_to_rlock_transitioned variable is constantly set to FFh.
- **The Optical_RT_present** variable is reset to 0b.
- The next state is Detect.Active after a 12 ms timeout or if Electrical Idle Exit is detected on any Lane. If the transition to this state is from Hot Reset, it is strongly recommended that the Downstream Port wait for 2 ms before enabling Electrical Idle Exit detection.

 ECN: Base 6.3
 Optical $\triangle \triangle$

4.2.7.1.2 Detect.Active §

- The Transmitter performs a Receiver Detection sequence on all un-configured Lanes that can form one or more Links (see § Section 8.4.5.7 for more information).

- Next state is Polling if a Receiver is detected on all unconfigured Lanes.
- Next state is Detect.Quiet if a Receiver is not detected on any Lane.
- If at least one but not all un-configured Lanes detect a Receiver, then:
 1. Wait for 12 ms.
 2. The Transmitter performs a Receiver Detection sequence on all un-configured Lanes that can form one or more Links (see § Section 8.4.5.7 for more information).
 - The next state is Polling if exactly the same Lanes detect a Receiver as the first Receiver Detection sequence.
 - Lanes that did not detect a Receiver must:
 - i. Be associated with a new LTSSM if this optional feature is supported.
or
 - ii. All Lanes that cannot be associated with an optional new LTSSM must transition to Electrical Idle.⁹⁰
 - These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.
 - An EIOS does not need to be sent before transitioning to Electrical Idle.
 - Otherwise, the next state is Detect.Quiet.

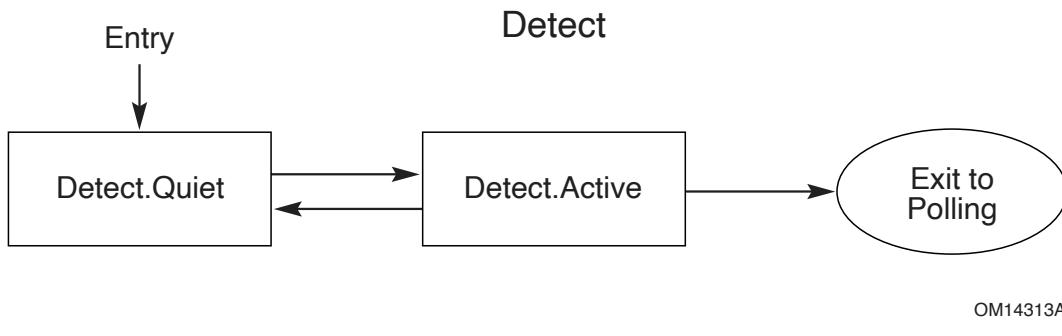


Figure 4-68 Detect Substate Machine §

4.2.7.2 Polling §

The Polling substate machine is shown in § Figure 4-69 .

4.2.7.2.1 Polling.Active §

- Transmitter sends TS1 Ordered Sets with Lane and Link numbers set to PAD on all Lanes that detected a Receiver during Detect.
 - The Data Rate Identifier Symbol of the TS1 Ordered Sets must advertise all data rates between 2.5 GT/s and 32.0 GT/ that the Port supports, including those that it does not intend to use.

Note: Ports are not permitted to advertise higher than 32.0 GT/s data rates in this state.

90. The common mode being driven is not required to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).

- Base 6.4 vs Base 6.3**
- The Transmitter must wait for its TX common mode to settle before exiting from Electrical Idle and transmitting the TS1 Ordered Sets.
 - The Transmitter must drive patterns in the default voltage level of the Transmit Margin field within 192 ns from entry to this state. This transmit voltage level will remain in effect until Polling.Compliance or Recovery.RcvrLock is entered.
 - Next state is Polling.Compliance if the Enter Compliance bit (bit 4) in the Link Control 2 Register is 1b. If the Enter Compliance bit was set prior to entry to Polling.Active, the transition to Polling.Compliance must be immediate without sending any TS1 Ordered Sets.
 - Next state is Polling.Configuration after at least 1024 TS1 Ordered Sets were transmitted, and all Lanes that detected a Receiver during Detect receive eight consecutive training sequences (or their complement) satisfying any of the following conditions:
 - TS1 with Lane and Link numbers set to PAD and Compliance_Receive_Request deasserted.
 - TS1 with Lane and Link numbers set to PAD and Loopback_Request asserted.
 - TS2 with Lane and Link numbers set to PAD.
 - Otherwise, after a 24 ms timeout the next state is:
 - Polling.Configuration if,
 - i. Any Lane, which detected a Receiver during Detect, received eight consecutive training sequences (or their complement) satisfying any of the following conditions:
 1. TS1 with Lane and Link numbers set to PAD and with Compliance_Receive_Request deasserted.
 2. TS1 with Lane and Link numbers set to PAD and Loopback_Request asserted.
 3. TS2 with Lane and Link numbers set to PAD.
- and a minimum of 1024 TS1 Ordered Sets are transmitted after receiving one TS1 or TS2 Ordered Set⁹¹.
- And
- ii. At least a predetermined set of Lanes that detected a Receiver during Detect have detected an exit from Electrical Idle at least once since entering Polling.Active.
 - Note: This may prevent one or more bad Receivers or Transmitters from holding up a valid Link from being configured, and allow for additional training in Polling.Configuration. The exact set of predetermined Lanes is implementation specific. Note that up to [PCIe-1.1] this predetermined set was equal to the total set of Lanes that detected a Receiver.
 - Note: Any Lane that receives eight consecutive TS1 or TS2 Ordered Sets should have detected an exit from Electrical Idle at least once since entering Polling.Active.
 - Else Polling.Compliance if either (a) or (b) is true:
 - a. not all Lanes from the predetermined set of Lanes from (ii) above have detected an exit from Electrical Idle since entering Polling.Active.
 - b. any Lane that detected a Receiver during Detect received eight consecutive TS1 Ordered Sets (or their complement) with the Lane and Link numbers set to PAD, Compliance_Receive_Request asserted, and Loopback_Request deasserted.

91. Earlier versions of this specification required transmission of 1024 TS1 Ordered Sets after receiving one TS1 Ordered Set. This behavior is still permitted but the implementation will be more robust if it follows the behavior of transmitting 1024 TS1 Ordered Sets after receiving one TS1 or TS2 Ordered Set.

- A port that is capable of transmitting at the 64.0 GT/s data rate may receive TS1 Ordered Sets with the Flit Mode Supported bit set to 1b and the Supported Link Speeds field set to 1 0111b.
- Note: If a passive test load is applied on all Lanes then the device will go to Polling.Compliance .
- Else Detect if the conditions to transition to Polling.Configuration or Polling.Compliance are not met

4.2.7.2.2 Polling.Compliance §

- The Transmit Margin field of the Link Control 2 Register is sampled on entry to this substate and becomes effective on the transmit package pins within 192 ns of entry to this substate and remains effective through the time the LTSSM is in this substate.
- The data rate and de-emphasis level for transmitting the compliance pattern are determined on the transition from Polling.Active to Polling.Compliance using the following algorithm.
 - If the Port is capable of transmitting at the 2.5 GT/s data rate only, the data rate for transmitting the compliance pattern is 2.5 GT/s and the de-emphasis level is -3.5 dB.
 - Else if the Port entered Polling.Compliance due to detecting eight consecutive TS1 Ordered Sets in Polling.Active with Compliance_Receive_Request asserted and Loopback_Request deasserted then the data rate for transmission is determined as follows:
 - If the port is capable of transmitting at the 64.0 GT/s data rate and the eight consecutive TS1 Ordered Sets received in Polling.Active that directed the port to this state had the Flit Mode Supported bit set to 1b and the Supported Link Speeds field set to 1 0111b, the data rate for transmitting the compliance pattern must be 64.0 GT/s.
 - Else, the data rate for transmission is determined by the highest common transmitted and received Data Rate Identifiers (Symbols 4 of the TS1 sequence) advertised in the eight consecutive TS1 Ordered Sets received on any Lane that detected a Receiver during Detect.

The select_deemphasis variable must be set equal to the Selectable De-emphasis bit (Symbol 4 bit 6) in the eight consecutive TS1 Ordered Sets it received in Polling.Active substate. If the common data rate is 8.0 GT/s or higher, the select_preset variable on each Lane is set to the Transmitter preset value advertised in the Transmitter Preset bits of the eight consecutive EQ TS1 Ordered Sets on the corresponding Lane, provided the value is not a Reserved encoding, and this value must be used by the transmitter (for 8.0 GT/s Data Rate, use of the Receiver preset hint value advertised in those eight consecutive EQ TS1 Ordered Sets is optional). If the common Data Rate is 8.0 GT/s or higher, any Lanes that did not receive eight consecutive EQ TS1 Ordered Sets with Transmitter preset information, or that received a value for a Reserved encoding, can use any supported Transmitter preset in an implementation specific manner.

- Else if the Enter Compliance bit in the Link Control 2 Register is 1b, the data rate for transmitting the compliance pattern is defined by the Target Link Speed field in the Link Control 2 Register . The select_deemphasis variable is Set when the Compliance Preset/De-emphasis field in the Link Control 2 Register equals 0001b if the data rate will be 5.0 GT/s. If the data rate will be 8.0 GT/s or higher, the select_preset variable on each Lane is set to, and the transmitter must operate with, the preset value provided in the Compliance Preset/De-emphasis Value (bits 15:12) in the Link Control 2 Register provided the value is not a Reserved encoding.
- Else the data rate, preset, and de-emphasis level settings are defined as follows based on the component's maximum supported data rate and the number of times Polling.Compliance has been entered with this entry criteria, in the same sequence of setting numbers as described in § Table 4-61 :

Table 4-61 Compliance Pattern Settings §

| Setting Nos | Data Rate | Transmitter De-emphasis or preset sequence |
|-----------------|-----------|--|
| #1 | 2.5 GT/s | -3.5 dB |
| #2, #3 | 5.0 GT/s | -3.5 dB followed by -6 dB |
| #4 through #14 | 8.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #15 through #25 | 16.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #26 through #34 | 16.0 GT/s | Transmitter Preset Encoding 0100b as defined in § Section 4.2.4.2 |
| #35 through #45 | 32.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #46 through #54 | 32.0 GT/s | Transmitter Preset Encoding 0100b as defined in § Section 4.2.4.2 |
| #55 through #65 | 64.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #66 through #84 | 64.0 GT/s | Transmitter Preset Encoding 0000b as defined in § Section 4.2.4.2 |

Subsequent entries to [Polling.Compliance](#) repeat the above sequence. For example, the state sequence which causes a Port to transmit the Compliance Pattern at a data rate of 5.0 GT/s and a de-emphasis level of -6 dB is: [Polling.Active](#), [Polling.Compliance](#) (2.5 GT/s and -3.5 dB), [Polling.Active](#), [Polling.Compliance](#) (5.0 GT/s and -3.5 dB), [Polling.Active](#), [Polling.Compliance](#) (5.0 GT/s and -6 dB).

The sequence must be set to Setting #1 in the [Polling.Configuration](#) state if the Port supports 16.0 GT/s or higher Data Rates, or the Port's Receivers do not meet the [Z_{RX-DC}](#) specification for 2.5 GT/s when they are operating at 8.0 GT/s or higher data rates (see § [Table 8-12](#)). All Ports are permitted to set the sequence to Setting #1 in the [Polling.Configuration](#) state.

IMPLEMENTATION NOTE: COMPLIANCE LOAD BOARD USAGE TO GENERATE COMPLIANCE PATTERNS §

It is envisioned that the compliance load (base) board may send a 100 MHz signal for about 1 ms on one leg of a differential pair at 350 mV peak-to-peak on any Lane to cycle the device to the desired speed and de-emphasis level. The device under test is required, based on its maximum supported data rate, to cycle through the following settings in order, for each entry to [Polling.Compliance](#) from [Polling.Active](#), starting with the first setting on the first entry to [Polling.Compliance](#) after the Fundamental Reset as defined in § [Table 4-61](#).

- If the compliance pattern data rate is not 2.5 GT/s and any [TS1 Ordered Sets](#) were transmitted in [Polling.Active](#) prior to entering [Polling.Compliance](#) :
 - The Transmitter sends two [Compliance TS1 Ordered Sets](#) if all of the following conditions are true:
 - The current data rate is 2.5 GT/s.

- Flit Mode is supported.
- The Transmitter is sequencing through one of the settings defined in § Table 4-61 .
- Transmitter sends either one EIOS or two consecutive EIOSs prior to entering Electrical Idle.
- If the compliance pattern data rate is not 2.5 GT/s and TS1 Ordered Sets were not transmitted in Polling.Active prior to entering Polling.Compliance , the Transmitter must enter Electrical Idle without transmitting any EIOSs. During the period of Electrical Idle, the data rate is changed to the new speed and stabilized. If the frequency of operation will be 5.0 GT/s, the de-emphasis/preset level must be set to -3.5 dB if the select_deemphasis variable is 1b else it must be set to -6 dB. If the frequency of operation will be 8.0 GT/s or higher, the Transmitter preset value must be set to the value in the select_preset variable.
- If the Transmitter enters Electrical Idle while in Polling.Compliance , the period of Electrical Idle must be greater than 1 ms but must not exceed 2 ms.
- Behavior during Polling.Compliance after the data rate and de-emphasis/preset level are determined must follow the following rules:
 - If the Port entered Polling.Compliance due to detecting eight consecutive TS1 Ordered Sets in Polling.Active with Compliance_Receive_Request asserted and Loopback_Request deasserted or both the Enter Compliance bit and the Enter Modified Compliance bit in the Link Control 2 Register are set to 1b then the Transmitter sends out the Modified Compliance Pattern at the above determined data rate with the error status Symbol set to all 0's on all Lanes that detected a Receiver during Detect .
 - If the data rate is 2.5 GT/s or 5.0 GT/s, a particular Lane's Receiver independently signifies a successful lock to the incoming Modified Compliance Pattern by looking for any one occurrence of the Modified Compliance Pattern and then setting the Pattern Lock bit (bit 7 of the error status Symbol) in the same Lane of its own transmitted Modified Compliance Pattern.
 - The error status Symbols are not to be used for the lock process since they are undefined at any given moment.
 - An occurrence is defined above as the following sequence of 8b/10b Symbols; K28.5 , D21.5, K28.5 , and D10.2 or the complement of each of the individual Symbols.
 - The device under test must set the Pattern Lock bit of the Modified Compliance Pattern it transmits at the Transmitter package pin(s) after successfully locking to the incoming Modified Compliance Pattern within 1 ms of receiving the Modified Compliance Pattern at its Receiver package pin(s).
 - If the data rate is 8.0 GT/s or higher: The Error_Status field is set to 00h on entry to this substate. Each Lane sets the Pattern Lock bit independently when it achieves Block Alignment as described in § Section 4.2.2.2.1 . After Pattern Lock is achieved, Symbols received in Data Blocks are compared to the Idle data Symbol (00h) and each mismatched Symbol causes the Receiver Error Count field to be incremented by 1. The Receiver Error Count saturates at 127 (further mismatched Symbols do not change the Receiver Error Count). The Pattern Lock and Receiver Error Count information for each Lane is transmitted as part of the SKP Ordered Sets transmitted in that Lane's Modified Compliance Pattern. See § Section 4.2.8 for more information. The device under test must set the Pattern Lock bit in the SKP Ordered Set it transmits within 4 ms of receiving the Modified Compliance Pattern at its Receiver package pin(s).
- The scrambling requirements defined in § Section 4.2.2.4 are applied to the received Modified Compliance Pattern. For example, the scrambling LFSR seed is set per Lane, an EIEOS initializes the LFSR and SKP Ordered Sets do not advance the LFSR.

IMPLEMENTATION NOTE: HANDLING BIT SLIP AND BLOCK ALIGNMENT §

Devices should ensure that their Receivers have stabilized before attempting to obtain Block alignment and signaling Pattern Lock. For example, if an implementation expects to see bit slips in the initial few bits, it should wait for that time to be over before settling on a Block Alignment. Devices may also want to revalidate their Block alignment prior to setting the Pattern Lock bit.

- If the data rate is 2.5 GT/s or 5.0 GT/s, once a particular Lane indicates it has locked to the incoming Modified Compliance Pattern the Receiver Error Count for that particular Lane is incremented every time a Receiver Error occurs.
 - The error status Symbol uses the lower 7 bits as the Receiver Error Count field and this field will remain stuck at all 1's if the count reaches 127.
 - The Receiver must not make any assumption about the 10-bit patterns it will receive when in this substate if 8b/10b encoding is used.
- If the Enter Compliance bit in the Link Control 2 Register is 0b, the next state is Detect if directed
- Else if the Enter Compliance bit was set to 1b on entry to Polling.Compliance, next state is Polling.Active if any of the following conditions apply:
 - The Enter Compliance bit in the Link Control 2 Register has changed to 0b
 - The Port is an Upstream Port and an EIOS is received on any Lane. The Enter Compliance bit is reset to 0b when this condition is true.

If the Transmitter was transmitting at a data rate other than 2.5 GT/s, or the Enter Compliance bit in the Link Control 2 Register was set to 1b during entry to Polling.Compliance, the Transmitter sends eight consecutive EIOS and enters Electrical Idle prior to transitioning to Polling.Active. During the period of Electrical Idle, the data rate is changed to 2.5 GT/s and stabilized and the de-emphasis level is set to -3.5 dB.

Note: Sending multiple EIOS provides enough robustness such that the other Port detects at least one EIOS and exits Polling.Compliance substate when the configuration register mechanism was used for entry.

- Else if the Port entered Polling.Compliance due to the Enter Compliance bit of the Link Control 2 Register being set to 1b and the Enter Modified Compliance bit of the Link Control 2 Register being set to 0b:
 - a. Transmitter sends out the compliance pattern on all Lanes that detected a Receiver during Detect at the data rate and de-emphasis/preset level determined above.
 - b. Next state is Polling.Active if any of the following two conditions are true:
 1. The Enter Compliance bit in the Link Control 2 Register has changed to 0b (from 1b) since entering Polling.Compliance.
 2. The Port is an Upstream Port, the Enter Compliance bit in the Link Control 2 Register is set to 1b and an EIOS has been detected on any Lane. The Enter Compliance bit is reset to 0b when this condition is true.

The Transmitter sends eight consecutive EIOSs and enters Electrical Idle prior to transitioning to Polling.Active. During the period of Electrical Idle, the data rate is changed to 2.5 GT/s and stabilized.

Note: Sending multiple EIOSs provides enough robustness such that the other Port detects at least one EIOS and exits Polling.

- Else:

- a. Transmitter sends out the following patterns on Lanes that detected a Receiver during Detect at the data rate and de-emphasis/preset level determined above:
 - For Settings #1 to #25, #35 to #45, and #55 to #65: Compliance Pattern on all Lanes.
 - For Setting #26, #46, #66: Jitter Measurement Pattern on all Lanes.
 - For Setting #27, #47, #67: Jitter Measurement Pattern on Lanes 0/8 and Compliance Pattern on all other Lanes.
 - For Setting #28, #48, #68: Jitter Measurement Pattern on Lanes 1/9 and Compliance Pattern on all other Lanes.
 - For Setting #29, #49, #69: Jitter Measurement Pattern on Lanes 2/10 and Compliance Pattern on all other Lanes.
 - For Setting #30, #50, #70: Jitter Measurement Pattern on Lanes 3/11 and Compliance Pattern on all other Lanes.
 - For Setting #31, #51, #71: Jitter Measurement Pattern on Lanes 4/12 and Compliance Pattern on all other Lanes.
 - For Setting #32, #52, #72: Jitter Measurement Pattern on Lanes 5/13 and Compliance Pattern on all other Lanes.
 - For Setting #33, #53, #73: Jitter Measurement Pattern on Lanes 6/14 and Compliance Pattern on all other Lanes.
 - For Setting #34, #54, #74: Jitter Measurement Pattern on Lanes 7/15 and Compliance Pattern on all other Lanes.
 - For Setting #75: High Swing Toggle Pattern on all Lanes
 - For Setting #76: High Swing Toggle Pattern on Lanes 0/8 and Compliance Pattern on all other Lanes.
 - For Setting #77: High Swing Toggle Pattern on Lanes 1/9 and Compliance Pattern on all other Lanes.
 - For Setting #78: High Swing Toggle Pattern on Lanes 2/10 and Compliance Pattern on all other Lanes.
 - For Setting #79: High Swing Toggle Pattern on Lanes 3/11 and Compliance pattern on all other Lanes.
 - For Setting #80: High Swing Toggle Pattern on Lanes 4/12 and Compliance Pattern on all other Lanes.
 - For Setting #81: High Swing Toggle Pattern on Lanes 5/13 and Compliance Pattern on all other Lanes.
 - For Setting #82: High Swing Toggle Pattern on Lanes 6/14 and Compliance Pattern on all other Lanes.
 - For Setting #83: High Swing Toggle Pattern on Lanes 7/15 and Compliance Pattern on all other Lanes.
 - For Setting #84: Low Swing Toggle Pattern on all Lanes
- b. Next state is Polling.Active if an exit of Electrical Idle is detected at the Receiver of any Lane that detected a Receiver during Detect. If the Transmitter is transmitting at a data rate other than 2.5 GT/s, the Transmitter sends eight consecutive EIOSs and enters Electrical Idle prior to transitioning to Polling.Active. During the period of Electrical Idle, the data rate is changed to 2.5 GT/s and stabilized.

4.2.7.2.3 Polling.Configuration §

- Receiver must invert polarity if necessary (see § [Section 4.2.5.5](#)).
- The Transmit Margin field of the [Link Control 2 Register](#) must be reset to 000b on entry to this substate.
- The Transmitter's [Polling.Compliance](#) sequence setting is updated, if required, as described in § [Section 4.2.7.2.2](#) .
- Transmitter sends [TS2 Ordered Sets](#) with Link and Lane numbers set to PAD on all Lanes that detected a Receiver during [Detect](#).
 - The Data Rate Identifier Symbol of the [TS2 Ordered Sets](#) must advertise all data rates between 2.5 GT/s and 32.0 GT/s that the Port supports, including those that it does not intend to use.

Note: Ports are not permitted to advertise higher than 32.0 GT/s data rates in this state.

- The next state is [Configuration](#) after eight consecutive [TS2 Ordered Sets](#), with Link and Lane numbers set to PAD, are received on any Lanes that detected a Receiver during [Detect](#), and 16 [TS2 Ordered Sets](#) are transmitted after receiving one [TS2 Ordered Set](#).
- Otherwise, next state is [Detect](#) after a 48 ms timeout.

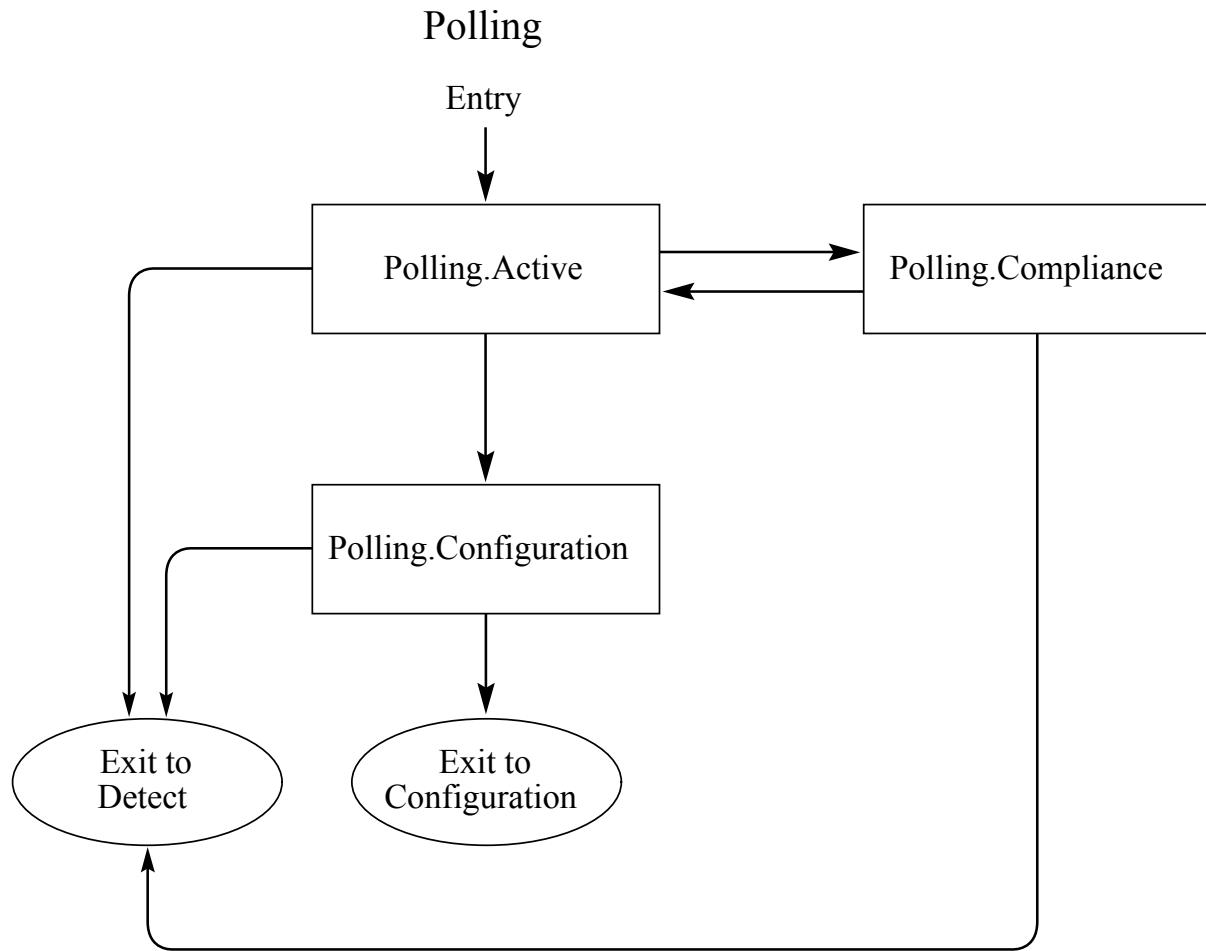
4.2.7.2.4 Polling.Speed §

This state is unreachable given that the Link comes up to [L0](#) in 2.5 GT/s data rate only and changes speed by entering [Recovery](#) .

IMPLEMENTATION NOTE: SUPPORT FOR HIGHER DATA RATES THAN 2.5 GT/S §

A Link will initially train to the [L0](#) state at the 2.5 GT/s data rate even if both sides are capable of operating at a data rate greater than 2.5 GT/s. Supported higher data rates are advertised in the [TS Ordered Sets](#). The other side's speed capability is registered during the [Configuration.Complete](#) substate. Based on the highest supported common data rate, either side can initiate a change in speed from the [L0](#) state by transitioning to [Recovery](#) .

Base 6.4 vs Base 6.3



OM13801B

Figure 4-69 Polling Substate Machine

4.2.7.3 Configuration

The Configuration substate machine is shown in § [Figure 4-70](#).

4.2.7.3.1 Configuration.Linkwidth.Start

4.2.7.3.1.1 Downstream Lanes

- Next state is Disabled if directed.
 - The clause “if directed” applies to a Downstream Port that is instructed by a higher Layer to assert Disable_Link_Request (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- Next state is Loopback if directed by an implementation specific method and the Transmitter is capable of being a Loopback Lead.
 - The clause “if directed” applies to a Port that is instructed by a higher Layer to assert Loopback_Request on all Lanes that detected a Receiver during Detect.

- In the optional case where a crosslink is supported, the next state is Disabled after all Lanes that are transmitting TS1 Ordered Sets receive two consecutive TS1 Ordered Sets with Disable_Link_Request asserted.
- Next state is Loopback if one of the following two conditions are satisfied:
 - All Lanes that are transmitting TS1 Ordered Sets, that are also receiving TS1 Ordered Sets, receive Loopback_Request asserted in two consecutive TS1 Ordered Sets.
 - Any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with Loopback_Request asserted and with the Enhanced Link Behavior Control bits set to 01b.

A Port that is capable of transmitting at the 64.0 GT/s data rate may receive TS1 Ordered Sets with the Flit Mode Supported bit set to 1b and the Supported Link Speeds field set to 1 0111b.

The device receiving the Ordered Set with Loopback_Request asserted becomes the Loopback Follower. If the Loopback Follower supports 1b/1b encoding, it must take into account whether SRIS clocking is enabled (Symbol 4, Bit 7) for identifying SKP OS boundary during Loopback.

- The Transmitter sends TS1 Ordered Sets with selected Link numbers and sets Lane numbers to PAD on all the active Downstream Lanes if LinkUp is 0b or if the LTSSM is not initiating upconfiguration of the Link width. In addition, if upconfigure_capable is set to 1b, and the LTSSM is not initiating upconfiguration of the Link width, the LTSSM sends TS1 Ordered Sets with the selected Link number and sets the Lane number to PAD on each inactive Lane after it detected an exit from Electrical Idle since entering Recovery and has subsequently received two consecutive TS1 Ordered Sets with the Link and Lane numbers each set to PAD while in this substate.
 - On transition to this substate from Polling, any Lane that detected a Receiver during Detect is considered an active Lane.
 - On transition to this substate from Recovery, any Lane that is part of the configured Link the previous time through Configuration.Complete is considered an active Lane.
 - The Data Rate Identifier Symbol of the TS1 Ordered Sets must advertise all data rates that the Port supports, including those that it does not intend to use.
- If LinkUp is 1b and the LTSSM is initiating upconfiguration of the Link width, initially it transmits TS1 Ordered Sets with both the Link and Lane numbers set to PAD on the current set of active Lanes; the inactive Lanes it intends to activate; and those Lanes where it detected an exit from Electrical Idle since entering Recovery and has received two consecutive TS1 Ordered Sets with the Link and Lane numbers each set to PAD. The LTSSM transmits TS1 Ordered Sets with the selected Link number and the Lane number set to PAD when each of the Lanes transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Link and Lane numbers each set to PAD or 1 ms has expired since entering this substate.
 - After activating any inactive Lane, the Transmitter must wait for its TX common mode to settle before exiting from Electrical Idle and transmitting the TS1 Ordered Sets.
 - Link numbers are only permitted to be different for groups of Lanes capable of being a unique Link.
 - Note: An example of Link number assignments is a set of eight Downstream Lanes capable of negotiating to become one x8 Port when connected to one component or two x4 Ports when connected to two different components. The Downstream Lanes send out TS1 Ordered Sets with the Link number set to N on four Lanes and Link number set to N+1 on the other four Lanes. The Lane numbers are all set to PAD.
- If any Lanes first received at least one or more TS1 Ordered Sets with a Link and Lane number set to PAD, the next state is Configuration.Linkwidth.Accept immediately after any of those same Downstream Lanes receive two consecutive TS1 Ordered Sets with a non-PAD Link number that matches any of the transmitted Link numbers, and with a Lane number set to PAD.
 - If the crosslink configuration is not supported, the condition of first receiving a Link and Lane number set to PAD is always true.

- Else: Optionally, if LinkUp is 0b and if crosslinks are supported, then all Downstream Lanes that detected a Receiver during Detect must first transmit 16 to 32 TS1 Ordered Sets with a non-PAD Link number and PAD Lane number and after this occurs if any Downstream Lanes receive two consecutive TS1 Ordered Sets with a Link number different than PAD and a Lane Number set to PAD, the Downstream Lanes are now designated as Upstream Lanes and a new random crosslink timeout is chosen (see $T_{crosslink}$ in § Table 8-7). The next state is Configuration.Linkwidth.Start as Upstream Lanes.
 - Note: This supports the optional crosslink where both sides may try to act as a Downstream Port. This is resolved by making both Ports become Upstream and assigning a random timeout until one side of the Link becomes a Downstream Port and the other side remains an Upstream Port. This timeout must be random even when hooking up two of the same devices so as to eventually break any possible deadlock.
 - If crosslinks are supported, receiving a sequence of TS1 Ordered Sets with a Link number of PAD followed by a Link number of non-PAD that matches the transmitted Link number is only valid when not interrupted by the reception of a TS2 Ordered Set.

IMPLEMENTATION NOTE: CROSSLINK INITIALIZATION §

In the case where the Downstream Lanes are connected to both Downstream Lanes (crosslink) and Upstream Lanes, the Port with the Downstream Lanes may continue with a single LTSSM as described in this section or optionally, split into multiple LTSSMs.

- The next state is Detect after a 24 ms timeout.

4.2.7.3.1.2 Upstream Lanes §

- In the optional case where crosslinks are supported the next state is Disabled if directed.
 - The clause “if directed” only applies to an optional crosslink Port that is instructed by a higher Layer to assert Disable_Link_Request (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- Next state is Loopback if directed to this state by an implementation specific method.
 - The clause “if directed” applies to a Port that is instructed by a higher Layer to assert Loopback_Request (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- Next state is Disabled after any Lanes that are transmitting TS1 Ordered Sets receive two consecutive TS1 Ordered Sets with Disable_Link_Request asserted.
 - In the optional case where a crosslink is supported, the next state is Disabled only after all Lanes that are transmitting TS1 Ordered Sets, that are also receiving TS1 Ordered Sets, receive Disable_Link_Request asserted in two consecutive TS1 Ordered Sets.
- Next state is Loopback if one of the following two conditions are satisfied:
 - All Lanes that are transmitting TS1 Ordered Sets, that are also receiving TS1 Ordered Sets, receive Loopback_Request asserted in two consecutive TS1 Ordered Sets.
 - Any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with Loopback_Request asserted and with the Enhanced Link Behavior Control bits set to 01b.

A Port that is capable of transmitting at the 64.0 GT/s data rate may receive TS1 Ordered Sets with the Flit Mode Supported bit set to 1b and the Supported Link Speeds field set to 1 0111b.

The device receiving the Ordered Set with Loopback_Request asserted becomes the Loopback Follower. If the Loopback Follower supports 1b/1b encoding, it must take into account whether SRIS Clocking is enabled (Symbol 4, Bit 7 = 1b) for identifying SKP OS boundaries during Loopback.

- The Transmitter sends out TS1 Ordered Sets with Link numbers and Lane numbers set to PAD on all the active Upstream Lanes; the inactive Lanes it is initiating to upconfigure the Link width; and if upconfigure_capable is set to 1b, on each of the inactive Lanes where it detected an exit from Electrical Idle since entering Recovery and has subsequently received two consecutive TS1 Ordered Sets with Link and Lane numbers, each set to PAD, in this substate.
 - On transition to this substate from Polling, any Lane that detected a Receiver during Detect is considered an active Lane.
 - On transition to this substate from Recovery, any Lane that is part of the configured Link the previous time through Configuration.Complete is considered an active Lane.
 - On transition to this substate from Recovery, if the transition is not caused by LTSSM timeout, the Transmitter must set the Autonomous Change bit (Symbol 4 bit 6) to 1b in the TS1 Ordered Sets that it sends while in the Configuration state if the Transmitter intends to change the Link width for autonomous reasons.
 - The Data Rate Identifier Symbol of the TS1 Ordered Sets must advertise all data rates that the Port supports, including those that it does not intend to use.
- If any Lane receives two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD, a single Link number is selected and Lane number set to PAD are transmitted on all Lanes that both detected a Receiver and also received two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD. Any left over Lanes that detected a Receiver during Detect must transmit TS1 Ordered Sets with the Link and Lane number set to PAD. The next state is Configuration.Linkwidth.Accept:
 - If the LTSSM is initiating upconfiguration of the Link width, it waits until it receives two consecutive TS1 Ordered Sets with a non-PAD Link Number and a PAD Lane number on all the inactive Lanes it wants to activate, or, 1 ms after entry to this substate, it receives two consecutive TS1 Ordered Sets on any Lane with a non-PAD Link number and PAD Lane number, whichever occurs earlier, before transmitting TS1 Ordered Sets with selected Link number and Lane number set to PAD.
 - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment or Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes; delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b or 1b/1b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.
 - After activating any inactive Lane, the Transmitter must wait for its TX common mode to settle before exiting Electrical Idle and transmitting the TS1 Ordered Sets.
- Optionally, if LinkUp is 0b and if crosslinks are supported, then all Upstream Lanes that detected a Receiver during Detect must first transmit 16-32 TS1 Ordered Sets with a PAD Link number and PAD Lane number and after this occurs and if any Upstream Lanes first receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD, then:
 - The Transmitter continues to send out TS1 Ordered Sets with Link numbers and Lane numbers set to PAD.
 - If any Lanes receive two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD, a single Link number is selected and Lane number set to PAD are transmitted on all Lanes that both detected a Receiver and also received two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD. Any left over Lanes that detected a Receiver during Detect must transmit TS1 Ordered Sets with the Link and Lane number set to PAD. The next state is Configuration.Linkwidth.Accept.

- It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment or Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes; delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b or 1b/1b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.
- Otherwise, after a $T_{crosslink}$ timeout, 16 to 32 TS2 Ordered Sets with PAD Link numbers and PAD Lane numbers are sent. The Upstream Lanes become Downstream Lanes and the next state is Configuration.Linkwidth.Start as Downstream Lanes.
 - Note: This optional behavior is required for crosslink behavior where two Ports may start off with Upstream Ports, and one will eventually take the lead as a Downstream Port.
- The next state is Detect after a 24 ms timeout.

4.2.7.3.2 Configuration.Linkwidth.Accept §

4.2.7.3.2.1 Downstream Lanes §

- If a configured Link can be formed with at least one group of Lanes that received two consecutive TS1 Ordered Sets with the same received Link number (non-PAD and matching one that was transmitted by the Downstream Lanes), TS1 Ordered Sets are transmitted with the same Link number and unique non-PAD Lane numbers are assigned to all these same Lanes. The next state is Configuration.Lanenum.Wait .
 - The assigned non-PAD Lane numbers must **↑↓adhere to the following:**
 - **↑↓Assigned Lane numbers must** range from 0 to **↑↓n-1,↓↑↓n-1,↑**
 - **↑↓Lane numbers must** be assigned sequentially to the same grouping of Lanes that are receiving the same Link **↑↓number, and↓↑↓number.↑**
 - Downstream Lanes which are not receiving **↑↓TS1↓↑↓TS1↑** Ordered Sets must not disrupt the initial sequential numbering of the widest possible Link. **↑↓Any left over↓**
 - **↑↓All remaining↓** Lanes must transmit **↑↓TS1↓↑↓TS1↑** Ordered Sets with the Link and Lane number set to PAD.
 - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment or Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b or 1b/1b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.
 - **↑↓At Data rates of 64.0 GT/s or higher, Downstream Lanes are permitted to delay up to 1 ms even if no block alignment loss or errors are detected. The reason for delaying up to 1 ms before transitioning is to prevent skew between Lanes from affecting the final configured Link width.↑**

Errata: Base 6.3
B825△↔Errata: Base 6.3
B840△↔

IMPLEMENTATION NOTE: POTENTIAL TO CONFIGURE LESS THAN MAXIMUM LINK WIDTH §

↑↑At 64.0 GT/s and above, $L_{rx-skew}$ values are large enough such that receivers can see greater than two TS1 Ordered Set Sequences worth of skew across all lanes. Even if no errors or block alignment loss is detected, it is possible that with a skew of greater than two TS1 Ordered Sets, a Link may be configured at less than the maximum possible width. To avoid this scenario, the consecutive TS1 Ordered Sets count must be qualified only after accounting for Lane skew in Configuration.Linkwidth.Accept .↑

Errata: Base 6.3
B840△◀▶

- The use_modified_TS1_TS2_Ordered_Set variable must be set to 1b if all of the following conditions are true:
 - LinkUp = 0b
 - The component had transmitted Modified TS1/TS2 Ordered Sets supported value (11b) in the Enhanced Link Behavior Control field in Symbol 5 of TS1 and TS2 Ordered Sets in Polling and Configuration states since entering the Polling State
 - The received eight consecutive TS2 Ordered Sets on all Lanes of the configured Link that could be formed (see 1st bullet of § Section 4.2.7.3.2.1) that were part of the group that caused the transition from Polling.Configuration to Configuration state had the Modified TS1/TS2 Ordered Sets supported value (11b) in the Enhanced Link Behavior Control field in Symbol 5 and 32.0 GT/s data rate is supported bit is Set to 1b in the received eight consecutive TS2 Ordered Sets
- The Flit_Mode_Enabled variable must be set to 1b if all of the following conditions are true:
 - LinkUp = 0b
 - The component had transmitted Flit Mode Supported bit in the ‘Data Rate Identifier’ field (Symbol 4, Bit 0) in the TS1 and TS2 Ordered Sets in Polling and Configuration states since entering the Polling State
 - The received eight consecutive TS2 Ordered Sets on all Lanes of the currently configured Link that caused the transition from Polling.Configuration to Configuration state had the Flit Mode Supported bit in the Data Rate Identifier field (Symbol 4, Bit 0) set to 1b in the received eight consecutive TS2 Ordered Sets
- The next state is Detect after a 2 ms timeout or if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.7.3.2.2 Upstream Lanes §

- If a configured Link can be formed using Lanes that transmitted a non-PAD Link number which are receiving two consecutive TS1 Ordered Sets with the same non-PAD Link number and any non-PAD Lane number, TS1 Ordered Sets are transmitted with the same non-PAD Link number and Lane numbers that, if possible, match the received Lane numbers or are different, if necessary (i.e., Lane reversed). The next state is Configuration.Lanenum.Wait .↑↑At data rates of 64.0 GT/s and greater, Upstream Lanes are permitted to delay up to 1 ms to prevent skew between lanes from affecting the final configured link width.↑

Errata: Base 6.3
B840△◀▶

Base 6.4 vs Base 6.3

- The received consecutive TS1 Ordered Sets may be either standard TS1 Ordered Sets or Modified TS1 Ordered Sets. Modified TS1 Ordered Sets will only be received if the conditions to Set the use_modified_TS1_TS2_Ordered_Set variable (as described below) are met.
- The newly assigned Lane numbers must adhere to the following:
 - Newly assigned Lane numbers must range from 0 to m-1, be
 - Newly assigned Lane numbers sequentially only to some continuous grouping of Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 Lane numbers Ordered Sets always disrupt a continuous grouping and must not be included in this grouping, must grouping).
 - Newly assigned Lane numbers include either Lane 0 or Lane n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest received Lane number (n-1). Remaining
 - All remaining Lanes must transmit TS1 Lane numbers Ordered Sets with Link and Lane numbers set to PAD.
- If any possible multi-Lane Link received an error in a TS1 Ordered Set, lost 128b/130b Block Alignment, or lost Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes, it is recommended to do the following to avoid configuring a smaller Link than is possible:
 - When using 8b/10b encoding, delay the evaluation listed above by an additional two or more TS1 Ordered Sets, but must not delay by more than 1 ms.
 - When using 128b/130b or 1b/1b encoding, delay the evaluation listed above by an additional 34 or more TS1 Ordered Sets, but must not delay by more than 1 ms.
- The use_modified_TS1_TS2_Ordered_Set variable must be set to 1b if all of the following conditions are true:
 - LinkUp = 0b
 - The component has transmitted Modified TS1/TS2 Ordered Sets supported value (11b) in the Enhanced Link Behavior Control field in Symbol 5 of all TS1 and TS2 Ordered Sets in Polling and Configuration states since entering the Polling State
 - The received eight consecutive TS2 Ordered Sets on all Lanes of the configured Link that could be formed (see 1st bullet of § Section 4.2.7.3.2.2) that were part of the group that caused the transition from Polling.Configuration to Configuration state had the Modified TS1/TS2 Ordered Sets supported value (11b) in the Enhanced Link Behavior Control field in Symbol 5 and 32.0 GT/s data rate is supported bit is Set to 1b in the received eight consecutive TS2 Ordered Sets
- The Flit_Mode_Enabled variable must be set to 1b if all of the following conditions are true:
 - LinkUp = 0b
 - The component had transmitted Flit Mode Supported bit in the ‘Data Rate Identifier’ field (Symbol 4, Bit 0) in the TS1 and TS2 Ordered Sets in Polling and Configuration states since entering the Polling State
 - The received eight consecutive TS2 Ordered Sets on all Lanes of the currently configured Link that caused the transition from Polling.Configuration to Configuration state had the Flit Mode Supported bit in the Data Rate Identifier field (Symbol 4, Bit 0) set to 1b in the received eight consecutive TS2 Ordered Sets

 Errata: Base 6.3
 B840△◀

IMPLEMENTATION NOTE: NEGOTIATING FLIT MODE AND THE MODIFIED TS1/TS2 USAGE §

Negotiating Flit Mode and the Modified TS1/TS2 usage is orthogonal (i.e., a device can advertise and negotiate support for neither, either, or both). This note applies to the Downstream and Upstream Ports.

- The next state is Detect after a 2 ms timeout or if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

IMPLEMENTATION NOTE: EXAMPLE CASES §

Notable examples related to the configuration of Downstream Lanes:

1. A x8 Downstream Port, which can be divided into two x4 Links, sends two different Link numbers on to two x4 Upstream Ports. The Upstream Ports respond simultaneously by picking the two Link numbers. The Downstream Port will have to choose one of these sets of Link numbers to configure as a Link, and leave the other for a secondary LTSSM to configure (which will ultimately happen in Configuration.Complete).
2. A x8 Downstream Port where only seven Lanes are receiving TS1 Ordered Sets with the same received Link number (non-PAD and matching one that was transmitted by the Downstream Lanes) and an eighth Lane, which is in the middle or adjacent to those same Lanes, is not receiving a TS1 Ordered Set. In this case, the eighth Lane is treated the same as the other seven Lanes and Lane numbering for a x8 Lane should occur as described above.

Notable examples related to the configuration of Upstream Lanes:

1. A x8 Upstream Port is presented with Lane numbers that are backward from the preferred numbering. If the optional behavior of Lane reversal is supported by the Upstream Port, the Upstream Port transmits the same Lane numbers back to the Downstream Port. Otherwise, the opposite Lane numbers are transmitted back to the Downstream Port, and it will be up to the Downstream Port to optionally fix the Lane ordering or exit Configuration .
Optional Lane reversal behavior is required to configure a Link where the Lane numbers are reversed and the Downstream Port does not support Lane reversal. Specifically, the Upstream Port Lane reversal will accommodate the scenario where the default Upstream sequential Lane numbering (0 to n-1) is receiving a reversed Downstream sequential Lane number (n-1 to 0).
2. A x8 Upstream Port is not receiving TS1 Ordered Sets on the Upstream Port Lane 0:
 - a. In the case where the Upstream Port can only support a x8 or x1 Link and the Upstream Port can support Lane reversal. The Upstream Port will assign a Lane 0 to only the received Lane 7 (received Lane number n-1) and the remaining seven Lanes must transmit TS1 Ordered Sets with Link and Lane numbers set to PAD.
 - b. In the case where the Upstream Port can only support a x8 or x1 Link and the Upstream Port cannot support Lane reversal. No Link can be formed and the Upstream Port will eventually timeout after 2 ms and exit to Detect .
3. An optional x8 Upstream crosslink Port, which can be divided into two x4 Links, is attached to two x4 Downstream Ports that present the same Link number, and each x4 Downstream Port presents Lane numbers simultaneously that were each numbered 0 to 3. The Upstream Port will have to choose one of these sets of Lane numbers to configure as a Link, and leave the other for a second pass through Configuration .

4.2.7.3.3 Configuration.Lanenum.Accept §

In this sub-state, if use_modified_TS1_TS2_Ordered_Set variable is set to 1b:

- Transmitter must send Modified TS1 Ordered sets instead of TS1 Ordered Sets

- Receiver must check for receipt of Modified TS1 Ordered Sets instead of TS1 Ordered Sets [Note: See § Section 4.2.5.1 for the definition of identical consecutive modified TS1 Ordered Sets.]

IMPLEMENTATION NOTE: DETECTION OF OPTICAL AWARE RETIMERS IN EXISTING COMPONENTS §

↑↑Components that do not support the detection of Optical Aware Retimers due to a lack of the Optical_RT_present variable or the Optical Retimer Presence Detect Supported and Optical Retimer Presence Detected bits fields will continue to interoperate with Optical Aware Retimer Solutions. However, the use of the sideband messaging mechanism defined in this specification will either not be available or will have to be enabled in an implementation specific manner.↑

ECN: Base 6.3
Optical△↔

4.2.7.3.3.1 Downstream Lanes §

- If two consecutive TS1 Ordered Sets are received with non-PAD Link and non-PAD Lane numbers that match all the non-PAD Link and non-PAD Lane numbers (or reversed Lane numbers if Lane reversal is optionally supported) that are being transmitted in Downstream Lane TS1 Ordered Sets, the next state is Configuration.Complete . If the use_modified_TS1_TS2_Ordered_Set variable is Set and an Alternate Protocol Negotiation is being performed, the transition to Configuration.Complete must be delayed for 10 µs or until the Downstream Port receives the Upstream Port's response to that protocol request (whichever happens first). See § Section 4.2.5.2 for Alternate Protocol Negotiation details. Note that Retimers are permitted to delay the transition to Configuration.Complete , as described in § Section 4.3.10 .
 - The SRIS_Mode_Enabled variable is Set to 1b if all of the following conditions are true:
 - LinkUp = 0b
 - The Port has been transmitting SRIS Clocking (Symbol 4, bit 7 = 1b) in the transmitted TS1 Ordered Sets since it entered Configuration state.
 - The Link Bandwidth Management Status and Link Autonomous Bandwidth Status bits of the Link Status Register must be updated as follows on a Link bandwidth change if the current transition to Configuration state was from the Recovery state:
 - a. If the bandwidth change was initiated by the Downstream Port due to reliability issues, the Link Bandwidth Management Status bit is Set.
 - b. Else if the bandwidth change was not initiated by the Downstream Port and the Autonomous Change bit (Symbol 4 bit 6) in two consecutive received TS1 Ordered Sets is 0b, the Link Bandwidth Management Status bit is Set.
 - c. Else the Link Autonomous Bandwidth Status bit is Set.
 - If the Flit_Mode_Enabled variable is 1b and N_FTS[7:6] is set to 10b in the received consecutive TS1 Ordered Sets that are directing the Port to Configuration.Complete , a PCIe Optical Retimer Solution is present. Set the Optical_RT_present variable to 1b and the Optical Retimer Presence Detected bit to 1b when the Optical Retimer Presence Detect Supported is Set.↑
 - The condition of Reversed Lane numbers is defined strictly as the Downstream Lane 0 receiving a TS1 Ordered Set with a Lane number equal to n-1 and the Downstream Lane n-1 receiving a TS1 Ordered Set with a Lane number equal to 0.

ECN: Base 6.3
Optical△↔

- If any possible multi-Lane Link received an error in a TS1 Ordered Set, lost 128b/130b Block Alignment, or lost Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes, it is recommended to do the following to avoid configuring a smaller Link than is possible:
- When using 8b/10b encoding, delay the evaluation listed above by an additional two or more TS1 Ordered Sets, but must not delay by more than 1 ms.
 - When using 128b/130b or 1b/1b encoding, delay the evaluation listed above by an additional 34 or more TS1 Ordered Sets, but must not delay by more than 1 ms.
- If a configured Link can be formed with any subset of the Lanes that receive two consecutive TS1 Ordered Sets with the same transmitted non-PAD Link numbers and any non-PAD Lane numbers, TS1 Ordered Sets are transmitted with the same non-PAD Link numbers and new Lane numbers assigned and the next state is Configuration.Lanenum.Wait .
- ~~↑↑The newly assigned transmitted Lane numbers must adhere to the following:~~
 - The newly assigned transmitted Lane numbers must range from 0 to ~~↑↓m-1, ↓↑m-1↑~~
 - ~~↑↑The newly assigned transmitted Lane numbers must~~ be assigned sequentially only to some continuous grouping of the Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 Ordered Sets always disrupt a continuous grouping and must not be included in this ~~↑↓grouping), ↓↑grouping).↑~~
 - ~~↑↑The newly assigned transmitted Lane numbers~~ must include either Lane 0 or Lane n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest received Lane number (n-1). ~~↑↓Any left over~~
 - ~~↑↑All remaining~~ Lanes must transmit TS1 Ordered Sets with the Link and Lane number set to PAD.
- If any possible multi-Lane Link received an error in a TS1 Ordered Set, lost 128b/130b Block Alignment, or lost Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes, it is recommended to do the following to avoid configuring a smaller Link than is possible:
- When using 8b/10b encoding, delay the evaluation listed above by an additional two or more TS1 Ordered Sets, but must not delay by more than 1 ms.
 - When using 128b/130b or 1b/1b encoding, delay the evaluation listed above by an additional 34 or more TS1 Ordered Sets, but must not delay by more than 1 ms.
- The next state is Detect if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

Errata: Base 6.3
B825 Δ ~~◀▶~~

4.2.7.3.3.2 Upstream Lanes §

- If two consecutive TS2 Ordered Sets are received with non-PAD Link and non-PAD Lane numbers that match all non-PAD Link and non-PAD Lane numbers that are being transmitted in Upstream Lane TS1 Ordered Sets, the next state is Configuration.Complete .

If the use_modified_TS1_TS2_Ordered_Set variable is Set, an Alternate Protocol Negotiation was performed, and the Downstream Port decided not to use any Alternate Protocol, the received TS2 Ordered Sets will have Modified TS Usage set to a value as defined in § Table 4-41 . See § Section 4.2.5.2 for Alternate Protocol Negotiation details.

The SRIS_Mode_Enabled variable is set to 1b if all of the following conditions are true:

- LinkUp = 0b

- Any configured Lane in the Port has the SRIS Clocking bit set (Symbol 4, bit 7 = 1b) in the two consecutive TS2 Ordered Sets

Note that Retimers are permitted to delay the transition to Configuration.Complete , as described in § Section 4.3.10 .

IMPLEMENTATION NOTE: CLOCKING WITH SRIS MODE VS. NON-SRIS MODE §

Clocking with SRIS mode vs. non-SRIS mode advertisement was added for Flit Mode support since there is no sync hdr in the 1b/1b encoding. It is advertised by the Downstream Port. However, it is defined as orthogonal to Flit Mode support. This note applies to the Downstream and Upstream Ports.

- If a configured Link can be formed with any subset of the Lanes that receive two consecutive TS1 Ordered Sets with the same transmitted non-PAD Link numbers and any non-PAD Lane numbers, TS1 Ordered Sets are transmitted with the same non-PAD Link numbers and new Lane numbers assigned and the next state is Configuration.Lanenum.Wait .
 - The newly assigned transmitted Lane numbers must adhere to the following:**
 - The newly assigned transmitted Lane numbers must range from 0 to $\downarrow\downarrow m-1, \uparrow\uparrow m-1$.
 - The newly assigned transmitted Lane numbers must** be assigned sequentially only to some continuous grouping of Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any $\downarrow\downarrow TS1$ $\uparrow\uparrow TS1$) Ordered Sets always disrupt a continuous grouping and must not be included in this $\downarrow\downarrow$ grouping, $\downarrow\downarrow$ grouping). \uparrow
 - The newly assigned transmitted Lane numbers** must include either Lane 0 or Lane n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest received Lane number (n-1). $\downarrow\downarrow$ Any left over
 - All remaining Lanes must transmit $\downarrow\downarrow TS1$ $\uparrow\uparrow TS1$ Ordered Sets with the Link and Lane number set to PAD.
 - If any possible multi-Lane Link received an error in a TS1 Ordered Set, lost 128b/130b Block Alignment, or lost Block/Flit alignment with 1b/1b encoding on a subset of the received Lanes, it is recommended to do the following to avoid configuring a smaller Link than is possible:
 - When using 8b/10b encoding, delay the evaluation listed above by an additional two or more TS1 Ordered Sets, but must not delay by more than 1 ms.
 - When using 128b/130b or 1b/1b encoding, delay the evaluation listed above by an additional 34 or more TS1 Ordered Sets, but must not delay by more than 1 ms.
 - If the Flit_Mode_Enabled variable is 1b and N_FTS[7:6] is set to 10b in the received consecutive TS2 Ordered Sets that are directing the Port to Configuration.Complete, a PCIe Optical Retimer Solution is present. Set the Optical_RT_present variable to 1b and the Optical Retimer Presence Detected bit to 1b when the Optical Retimer Presence Detect Supported is Set.**
 - The next state is Detect if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

Errata: Base 6.3
B825△↔

FCN: Base 6.3
Optical△↔

4.2.7.3.4 Configuration.Lanenum.Wait §

In this sub-state, if use_modified_TS1_TS2_Ordered_Set variable is set to 1b:

- Transmitter must send Modified TS1 Ordered Sets instead of TS1 Ordered Sets
- Receiver must check for receipt of Modified TS1 Ordered Sets instead of TS1 Ordered Sets though it may receive TS1 Ordered Sets initially while the Link partner is transitioning to this sub-state [Note: These must be identical consecutive Modified TS1 Ordered Sets with valid parity in the last Symbol]

4.2.7.3.4.1 Downstream Lanes §

- The next state is Configuration.Lanenum.Accept if any of the Lanes that detected a Receiver during Detect receive two consecutive TS1 Ordered Sets which have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait, and not all the Lanes' Link numbers are set to PAD or two consecutive TS1 Ordered Sets have been received on all Lanes, with Link and Lane numbers that match what is being transmitted on all Lanes.

The Upstream Lanes are permitted to delay up to 1 ms before transitioning to Configuration.Lanenum.Accept.

The reason for delaying up to 1 ms before transitioning is to prevent received errors or skew between Lanes affecting the final configured Link width.

The condition of requiring reception of any Lane number different from when the Lane(s) first entered Configuration.Lanenum.Wait is necessary in order to allow the two Ports to settle on an agreed upon Link width. The exact meaning of the statement “any of the Lanes receive two consecutive TS1 Ordered Sets, which have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait” requires that a Lane number must have changed from when the Lanes most recently entered Configuration.Lanenum.Wait before a transition to Configuration.Lanenum.Accept can occur.

- The next state is Detect after a 2 ms timeout or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.7.3.4.2 Upstream Lanes §

- The next state is Configuration.Lanenum.Accept
 - If any of the Lanes receive two consecutive TS1 Ordered Sets that have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait, and not all the Lanes' Link numbers are set to PAD

or
 - If any Lane receives two consecutive TS2 Ordered Sets
- The next state is Detect after a 2 ms timeout or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.7.3.5 Configuration.Complete §

A Port is allowed to change the supported data rates that it advertises when it enters this substate, but it must not change those values while in this substate.

If Flit_Mode_Enabled is 0b and LinkUp =1b, a Port is permitted to change the Upconfigure that it advertises when it enters this substate, but it must not change the value while in this substate.

If Flit_Mode_Enabled is 1b and LinkUp =0b, a Port is permitted to change the L0p Capability that it advertises when it enters this substate, but it must not change the value while in this substate.

In this sub-state, if use_modified_TS1_TS2_Ordered_Set variable is set to 1b:

- Transmitter must send Modified TS2 Ordered sets instead of TS2 Ordered Sets
- Receiver must check for receipt of Modified TS2 Ordered Sets, instead of TS2 Ordered Sets [Note: See § Section 4.2.5.1 for the definition of identical consecutive Modified TS1 Ordered Sets.]

4.2.7.3.5.1 Downstream Lanes §

- TS2 Ordered Sets are transmitted using Link and Lane numbers that match the received TS1 Ordered Set Link and Lane numbers.
 - The Link Upconfigure / L0p Capability bit of the TS2 Ordered Sets is permitted to be set to 1b to indicate that the Port is capable of supporting down to a x1 Link on the currently assigned Lane 0 and up-configuring the Link while LinkUp = 1b. Advertising this capability is optional.
- N_FTS must be noted for use in L0s when leaving this state.
- When using 8b/10b encoding, Lane-to-Lane de-skew must be completed when leaving this state.
- Scrambling is disabled if all configured Lanes have the Disable Scrambling bit asserted in two consecutively received TS2 Ordered Sets.
 - The Port that is sending the Disable Scrambling bit on all of the configured Lanes must also disable scrambling. Scrambling can only be disabled when using 8b/10b encoding.
- The next state is Configuration.Idle immediately after all Lanes that are transmitting TS2 Ordered Sets receive eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD) and identical data rate identifiers (including identical Link Upconfigure / L0p Capability bit (Symbol 4 bit 6)), and 16 TS2 Ordered Sets are sent after receiving one TS2 Ordered Set. Implementations with the Retimer Presence Detect Supported bit of the Link Capabilities 2 Register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bit 4) when the data rate is 2.5 GT/s. Implementations with Two Retimers Presence Detect Supported bit of the Link Capabilities 2 Register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bits 5:4) when the data rate is 2.5 GT/s.
 - If the data rate of operation is 2.5 GT/s:
 - If the Retimer Presence Detect Supported bit of the Link Capabilities 2 Register is set to 1b and any configured Lane received the Retimer Present bit set to 1b in the eight consecutively received TS2 Ordered Sets, then the Retimer Presence Detected bit must be set to 1b in the Link Status 2 Register otherwise the Retimer Presence Detected bit must be set to 0b.
 - If the Two Retimers Presence Detect Supported bit of the Link Capabilities 2 Register is set to 1b and any configured Lane received the Two Retimers Present bit set to 1b in the eight consecutively received TS2 Ordered Sets then the Two Retimers Presence Detected bit must be set to 1b in the Link Status 2 Register, otherwise the Two Retimers Presence Detected bit must be set to 0b.
 - If the device supports greater than 2.5 GT/s data rate, it must record the data rate identifier received on any configured Lane of the Link. This will override any previously recorded value. A variable to track speed change in recovery state, **changed_speed_recovery**, is reset to 0b.
 - If Flit_Mode_Enabled is 0b:

If the device sends TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit (Symbol 4 bit 6) set to 1b, and receives eight consecutive TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit set to 1b, the variable upconfigure_capable is set to 1b, else it is reset to 0b.
 - If Flit_Mode_Enabled is 1b and LinkUp=0b:

If the device sends TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit (Symbol 4 bit 6) set to 1b, and receives eight consecutive TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit set to 1b:

- The variable L0p_capable is set to 1b.
- The Remote L0p Supported bit in the Device Status 3 Register is set to 1b.
- All remaining Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must:
 - i. Be associated with a new LTSSM if this optional feature is supported.
or
 - ii. All Lanes that cannot be associated with an optional new LTSSM must transition to Electrical Idle.⁹² Those Lanes that formed a Link up to the L0 state, and LinkUp has been 1b since then, but are not a part of the currently configured Link, must be associated with the same LTSSM if the LTSSM advertises Link width upconfigure capability. It is recommended that the Receiver terminations of these Lanes be left on. If they are not left on, they must be turned on when the LTSSM enters the Recovery.RcvrCfg substate until it reaches the Configuration.Complete substate if upconfigure_capable is set to 1b to allow for potential Link width upconfiguration. Any Lane that was not part of the LTSSM during the initial Link training through L0 cannot become a part of the LTSSM as part of the Link width upconfiguration process.
 - In the case of an optional crosslink, the Receiver terminations are required to meet Z_{RX-HIGH-IMP-DC-POS} and Z_{RX-HIGH-IMP-DC-NEG} (see § Table 8-12).
 - These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect .
 - An EIOS does not need to be sent before transitioning to Electrical Idle, and the transition to Electrical Idle does not need to occur on a Symbol or Ordered Set boundary.
- After a 2 ms timeout:
 - The next state is Detect if the current data rate is 2.5 GT/s or 5.0 GT/s.
 - The next state is Configuration.Idle if the idle_to_rlock_transitions variable is less than FFh and the current data rate is 8.0 GT/s or higher.
 - i. The changed_speed_recovery variable is reset to 0b.
 - ii. Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must meet requirement (i) or (ii) specified above for the non-timeout transition to Configuration.Idle .
 - iii. The upconfigure_capable variable is permitted, but not required, to be updated if at least one Lane received eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD). If updated, the upconfigure_capable variable is set to 1b when the transmitted and received Link Upconfigure bits are 1b, else it is reset to 0b.
 - Else the next state is Detect .

92. The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).

4.2.7.3.5.2 Upstream Lanes §

- TS2 Ordered Sets are transmitted using Link and Lane numbers that match the received TS2 Link and Lane numbers.
 - The Link Upconfigure / L0p Capability bit of the TS2 Ordered Sets is permitted to be set to 1b to indicate that the Port is capable of supporting a x1 Link on the currently assigned Lane 0 and up-configuring the Link while LinkUp = 1b. Advertising this capability is optional.
- N_FTS must be noted for use in L0s when leaving this state.
- When using 8b/10b encoding, Lane-to-Lane de-skew must be completed when leaving this state.
- Scrambling is disabled if all configured Lanes have the Disable Scrambling bit asserted in two consecutively received TS2 Ordered Sets.
 - The Port that is sending the Disable Scrambling bit on all of the configured Lanes must also disable scrambling. Scrambling can only be disabled when using 8b/10b encoding.
- The next state is Configuration.Idle immediately after all Lanes that are transmitting TS2 Ordered Sets receive eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD) and identical data rate identifiers (including identical Link Upconfigure / L0p Capability bit (Symbol 4 bit 6)), and 16 consecutive TS2 Ordered Sets are sent after receiving one TS2 Ordered Set. Implementations with the Retimer Presence Detect Supported bit of the Link Capabilities 2 Register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bit 4) when the data rate is 2.5 GT/s. Implementations with Two Retimers Presence Detect Supported bit of the Link Capabilities 2 Register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bits 5:4) when the data rate is 2.5 GT/s.
 - If the data rate of operation is 2.5 GT/s:
 - If the Retimer Presence Detect Supported bit of the Link Capabilities 2 Register is set to 1b and any configured Lane received the Retimer Present bit set to 1b in the eight consecutively received TS2 Ordered Sets, then the Retimer Presence Detected bit must be set to 1b in the Link Status 2 Register otherwise the Retimer Presence Detected bit must be set to 0b.
 - If the Two Retimers Presence Detect Supported bit of the Link Capabilities 2 Register is set to 1b and any configured Lane received the Two Retimers Present bit set to 1b in the eight consecutively received TS2 Ordered Sets then the Two Retimers Presence Detected bit must be set to 1b in the Link Status 2 Register, otherwise the Two Retimers Presence Detected bit must be set to 0b.
 - If the device supports greater than 2.5 GT/s data rate, it must record the data rate identifier received on any configured Lane of the Link. This will override any previously recorded value. A variable to track speed change in recovery state, changed_speed_recovery, is reset to 0b.
 - If Flit_Mode_Enabled is 0b:

If the device sends TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit (Symbol 4 bit 6) set to 1b, as well as receives eight consecutive TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit set to 1b, the variable upconfigure_capable is set to 1b, else it is reset to 0b.
 - If Flit_Mode_Enabled is 1b and LinkUp=0b:

If the device sends TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit (Symbol 4 bit 6) set to 1b, and receives eight consecutive TS2 Ordered Sets with the Link Upconfigure / L0p Capability bit set to 1b:

 - The variable L0p_capable is set to 1b.
 - The Remote L0p Supported bit in the Device Status 3 Register is set to 1b.

- All remaining Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must:
 - i. Optionally be associated with a new crosslink LTSSM if this feature is supported.
or
 - ii. All remaining Lanes that are not associated with a new crosslink LTSSM must transition to Electrical Idle,⁹³ and Receiver terminations are required to meet Z_RX-HIGH-IMP-DC-POS and Z_RX-HIGH-IMP-DC-NEG (see § Table 8-12). Those Lanes that formed a Link up to the L0 state, and LinkUp has been 1b since then, but are not a part of the currently configured Link, must be associated with the same LTSSM if the LTSSM advertises Link width upconfigure capability. It is recommended that the Receiver terminations of these Lanes be left on. If they are not left on, they must be turned on when the LTSSM enters the Recovery.RcvrCfg substate until it reaches the Configuration.Complete substate if upconfigure_capable is set to 1b to allow for potential Link width upconfiguration. Any Lane that was not part of the LTSSM during the initial Link training through L0 cannot become a part of the LTSSM as part of the Link width upconfiguration process.
 - These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.
 - EIOS does not need to be sent before transitioning to Electrical Idle, and the transition to Electrical Idle does not need to occur on a Symbol or Ordered Set boundary.
- After a 2 ms timeout:
 - The next state is Detect if the current data rate is 2.5 GT/s or 5.0 GT/s.
 - The next state is Configuration.Idle if the idle_to_rlock_transitions variable is less than FFh and the current data rate is 8.0 GT/s or higher.
 - i. The changed_speed_recovery variable is reset to 0b.
 - ii. Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must meet requirement (i) or (ii) specified above for the non-timeout transition to Configuration.Idle.
 - iii. The upconfigure_capable variable is permitted, but not required, to be updated if at least one Lane received eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD). If updated, the upconfigure_capable variable is set to 1b when the transmitted and received Link Upconfigure bits are 1b, else it is reset to 0b.
 - Else the next state is Detect.

4.2.7.3.6 Configuration.Idle §

- When using 8b/10b encoding, the Transmitter sends Idle data Symbols on all configured Lanes in Non-Flit Mode and IDLE Flits across all configured Lanes in Flit Mode.
- If LinkUp = 0b and 64.0 GT/s data rate is supported by all components in the Link, as advertised in the eight consecutive TS2 or eight consecutive and identical Modified TS2 Ordered Sets received prior to entering Configuration.Idle:
 - If the No Equalization Needed bit (bit 1 of Symbol 5) was set to 1b in the received eight consecutive and identical Modified TS2 Ordered Sets and in the transmitted Modified TS2 Ordered Sets in all the configured Lanes of the Link or if No Equalization Needed value (10b) was received in the Enhanced

93. The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).

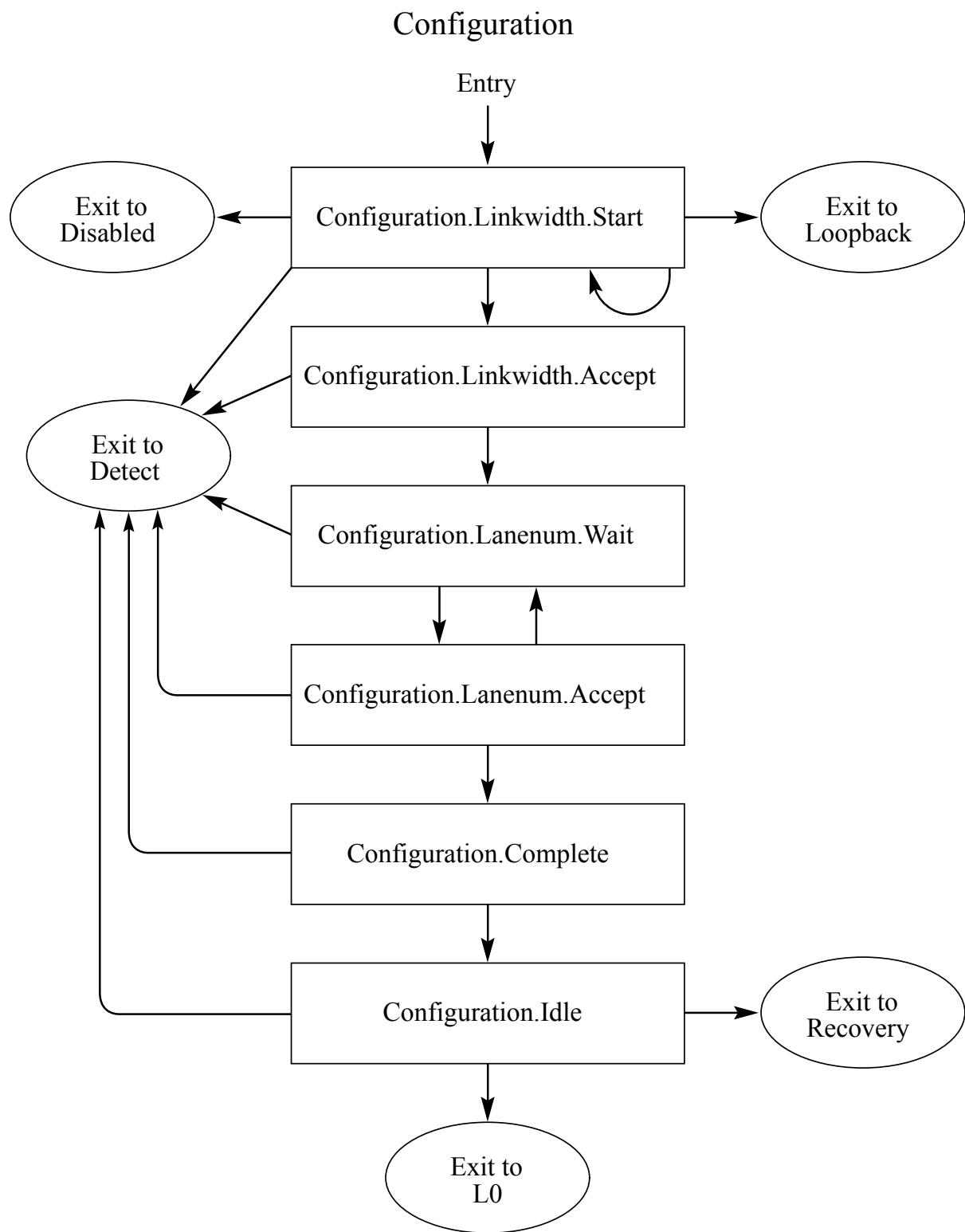
Link Behavior Control field (bits 7:6 of Symbol 5) in the eight consecutive TS2 Ordered Sets and was also set in the Enhanced Link Behavior Control field of the transmitted TS2 Ordered Sets:

- The equalization_done_8GT_data_rate, equalization_done_16GT_data_rate, equalization_done_32GT_data_rate, and equalization_done_64GT_data_rate variables are each set to 1b.
- The No Equalization Needed Received bit in the 64.0 GT/s Status Register is set to 1b.
- Else If the Equalization Bypass to Highest NRZ Rate bit (bit 0 of Symbol 5) was set to 1b in the received eight consecutive and identical Modified TS2 Ordered Sets as well as in the transmitted Modified TS2 Ordered Sets in all the configured Lanes of the Link or if either No Equalization Needed or Equalization Bypass to Highest NRZ Rate value (01b or 10b) was received in the Enhanced Link Behavior Control field (bits 7:6 of Symbol 5) in the eight consecutive TS2 Ordered Sets and either No Equalization Needed or Equalization Bypass to Highest NRZ Rate value (01b or 10b) was also set in the Enhanced Link Behavior Control field of the transmitted TS2 Ordered Sets:
 - The equalization_done_8GT_data_rate and equalization_done_16GT_data_rate variables are each set to 1b.
- If entry to this sub-state was caused by receipt of eight consecutive and identical Modified TS2 Ordered Sets and LinkUp = 0b
 - If the Modified TS Usage field in the received eight consecutive Modified TS2 Ordered Sets was set to 010b (Alternate Protocols) and the same value was set in the Modified TS Usage field of the transmitted Modified TS2 Ordered Sets and the Modified TS Information 1 and Alternate Protocol Vendor ID fields are identical between the transmitted and received Modified TS2 Ordered Sets in all the configured Lanes of the Link:
 - The Modified TS Received bit in the 32.0 GT/s Status Register is set to 1b. The details of the negotiation will be reflected in the Received Modified TS Data 1 Register and Received Modified TS Data 2 Register based on the eight consecutive Modified TS2 Ordered Sets received.
 - The equalization_done_8GT_data_rate, equalization_done_16GT_data_rate, and equalization_done_32GT_data_rate variables are each set to 1b.
 - The No Equalization Needed Received bit in the 32.0 GT/s Status Register is set to 1b.
- If LinkUp = 0b and 32.0 GT/s data rate is supported by all components in the Link, as advertised in the eight consecutive TS2 or eight consecutive and identical Modified TS2 Ordered Sets received prior to entering Configuration.Idle :
 - If the No Equalization Needed bit (bit 1 of Symbol 5) was set to 1b in the received eight consecutive and identical Modified TS2 Ordered Sets and was also set in the transmitted Modified TS2 Ordered Sets in all the configured Lanes of the Link or if No Equalization Needed value (10b) was received in the Enhanced Link Behavior Control field (bits 7:6 of Symbol 5) in the eight consecutive TS2 Ordered Sets and was also set in the Enhanced Link Behavior Control field of the transmitted TS2 Ordered Sets:
 - The equalization_done_8GT_data_rate, equalization_done_16GT_data_rate, and equalization_done_32GT_data_rate variables are each set to 1b.
 - The No Equalization Needed Received bit in the 32.0 GT/s Status Register is set to 1b.
 - Else If the Equalization Bypass to Highest NRZ Rate bit (bit 0 of Symbol 5) was set to 1b in the received eight consecutive and identical Modified TS2 Ordered Sets and was also set in the transmitted Modified TS2 Ordered Sets in all the configured Lanes of the Link or if either No Equalization Needed or Equalization Bypass to Highest NRZ Rate value (01b or 10b) was received in the Enhanced Link Behavior Control field (bits 7:6 of Symbol 5) in the eight consecutive TS2 Ordered Sets and either No Equalization Needed or Equalization Bypass to Highest NRZ Rate value (01b or 10b) was also set in the Enhanced Link Behavior Control field of the transmitted TS2 Ordered Sets:
 - The equalization_done_8GT_data_rate and equalization_done_16GT_data_rate variables are each set to 1b.
- If entry to this sub-state was caused by receipt of eight consecutive and identical Modified TS2 Ordered Sets and LinkUp = 0b

- If the Modified TS Usage field in the received eight consecutive Modified TS2 Ordered Sets was set to 010b (Alternate Protocols) and the same value was set in the Modified TS Usage field of the transmitted Modified TS2 Ordered Sets and the Modified TS Information 1 and Alternate Protocol Vendor ID fields are identical between the transmitted and received Modified TS2 Ordered Sets in all the configured Lanes of the Link:
 - The Modified TS Received bit in the 32.0 GT/s Status Register is set to 1b. The details of the negotiation will be reflected in the Received Modified TS Data 1 Register and Received Modified TS Data 2 Register based on the eight consecutive modified TS2 Ordered Sets received.
- When using 128b/130b encoding in Non-Flit Mode:
 - If the data rate is 8.0 GT/s, the Transmitter sends one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
 - If the data rate is 16.0 GT/s or higher, the Transmitter sends one Control SKP Ordered Set followed immediately by one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
- When using 1b/1b encoding or 128b/130b encoding in Flit Mode:
 - Transmitter sends an SDS Ordered Set sequence followed by one Control SKP Ordered Set on all configured Lanes to start a Data Stream with IDLE Flits .
- Receiver waits for Idle data in Non-Flit Mode and for IDLE Flits in Flit Mode.
- LinkUp = 1b
- When using 8b/10b encoding in Non-Flit Mode, the next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes and 16 Idle data Symbols are sent after receiving one Idle data Symbol.
 - If software has written a 1b to the Retrain Link bit in the Link Control Register since the last transition to L0 from Recovery or Configuration , the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status Register to 1b.
 - The use_modified_TS1_TS2_Ordered_Set variable is reset to 0b on transition to L0 .
- When using 128b/130b encoding in Non-Flit Mode, next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes, 16 Idle data Symbols are sent after receiving one Idle data Symbol, and this state was not entered by a timeout from Configuration.Complete .
 - The Idle data Symbols must be received in Data Blocks.
 - Lane-to-Lane de-skew must be completed before Data Stream processing starts.
 - If software has written a 1b to the Retrain Link bit in the Link Control Register since the last transition to L0 from Recovery or Configuration , the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status Register to 1b.
 - The idle_to_rlock_transitioned variable is reset to 00h on transition to L0 .
- In Flit Mode, the next state is L0 if two consecutive IDLE Flits are received and the minimum number of IDLE Flits are sent after receiving one IDLE Flit and this state was not entered by a timeout from Configuration.Complete. The minimum number of IDLE Flits to send is 4 with 8b/10b or 128b/130b encoding and 8 with 1b/1b encoding.
 - Lane-to-Lane de-skew must be completed before Data Stream processing starts.
 - If software has written a 1b to the Retrain Link bit in the Link Control Register since the last transition to L0 from Recovery or Configuration , the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status Register to 1b.

- The idle_to_rlock_transited variable is reset to 00h and the use_modified_TS1_TS2_Ordered_Set variable is reset to 0b on transition to L0 .
- Otherwise, after a minimum 2 ms timeout:
 - If the idle_to_rlock_transited variable is less than FFh, the next state is Recovery.RcvrLock .
 - On transition to Recovery.RcvrLock :
 - If the data rate is 8.0 GT/s or higher, the idle_to_rlock_transited variable is incremented by 1.
 - If the data rate is 2.5 GT/s or 5.0 GT/s, the idle_to_rlock_transited variable is set to FFh and the use_modified_TS1_TS2_Ordered_Set variable is reset to 0b.
 - Else the next state is Detect .

Base 6.4 vs Base 6.3



OM13802C

Figure 4-70 Configuration Substate Machine §

4.2.7.4 Recovery §

The Recovery substate machine is shown in § Figure 4-71 . For the data rate of 64.0 GT/s, any reference to Pre-cursor implies the two pre-cursors used.

4.2.7.4.1 Recovery.RcvrLock §

If the Link is operating at a data rate of 8.0 GT/s or higher, a Receiver must consider any TS0 , TS1 , or TS2 Ordered Set to be received only after it obtains Block Alignment in that Lane. If entry to this substate is from L1 or Recovery.Speed or L0s , the Block Alignment must be obtained after exiting Electrical Idle condition. If entry to this substate is from L0 , the Block Alignment must be obtained after the end of the last Data Stream.

- If the data rate of operation is 8.0 GT/s or higher:
 - If the start_equalization_w_preset variable is set to 1b:
 - An Upstream Port must use the Transmitter preset values it registered from the received appropriate eight consecutive TS2 Ordered Sets (EQ TS2 if 8.0 GT/s, EQ TS2 if 32.0 GT/s and Equalization Bypass to Highest NRZ Rate was negotiated, and 128b/130b EQ TS2 if 16.0 GT/s, 32.0 GT/s, or 64.0 GT/s) in Recovery.RcvrCfg in its Transmitter setting as soon as it starts transmitting in the data rate at which equalization will be performed and ensure that it meets the preset definition in § Chapter 8 . Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon as it starts transmitting at the data rate where equalization needs to be performed.
 - A Downstream Port must use the Transmitter preset settings according to the rules below as soon as it starts transmitting at the data rate where equalization must be performed:
 1. If the data rate of equalization is 16.0 GT/s or 32.0 GT/s or 64.0 GT/s and eight consecutive EQ TS2 Ordered Sets (for the case where equalization bypass to 32.0 GT/s is to be performed) or 128b/130b EQ TS2 Ordered Sets were received with supported Transmitter Preset values in the most recent transition through Recovery.RcvrCfg , the Transmitter Preset value from those EQ TS2 or 128b/130b EQ TS2 Ordered Sets must be used.
 2. Else, if the Transmitter preset value defined in the Downstream Port Transmitter Preset field of the appropriate Lane Equalization Control Register Entry , as defined below is supported, then it must be used:

| Data Rate of Equalization | Transmitter Preset value to be used as soon as the Link transitions to the data rate of equalization |
|---------------------------|--|
| 8.0 GT/s | Transmitter Preset field defined in the Lane Equalization Control Register Entry for each Lane. The Downstream Port may optionally use the Downstream Port 8.0 GT/s Receiver Preset Hint field defined in the Lane Equalization Control Register Entry for each of its Receivers corresponding to the Lane, if they are not Reserved values. |
| 16.0 GT/s | Downstream Port 16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register Entry |
| 32.0 GT/s | Downstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry |

| | |
|---------------------------|--|
| Data Rate of Equalization | Transmitter Preset value to be used as soon as the Link transitions to the data rate of equalization |
| 64.0 GT/s | Downstream Port 64.0 GT/s Transmitter Preset field of the 64.0 GT/s Lane Equalization Control Register Entry |

3. Else, use an implementation specific method to choose a supported Transmitter preset setting.

The Downstream Port must ensure that it meets the preset definition in § Chapter 8..

- Next state is Recovery.Equalization .
- Else:
 - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure.
 - If this substate was entered from Recovery.Equalization , in the transmitted TS1 Ordered Sets, a Downstream Port must set the Pre-cursor, Cursor, and Post-cursor Coefficient fields to the current Transmitter settings, and if the last accepted request in Phase 2 of Recovery.Equalization was a preset request, it must set the Transmitter Preset bits to the accepted preset of that request.
 - It is recommended that in this substate, in the transmitted TS1 Ordered Sets, all Ports set the Pre-cursor, Cursor, and Post-cursor Coefficient fields to the current Transmitter settings, and set the Transmitter Preset bits to the most recent preset that the Transmitter settings were set to.
 - An Upstream Port that receives eight consecutive TS0 or eight consecutive TS1 Ordered Sets on all configured Lanes with the following characteristics must transition to Recovery.Equalization
 - If eight consecutive TS1 Ordered Sets were received, Link and Lane numbers in the received TS1 Ordered Sets match with the Link and Lane numbers in the transmitted TS1 Ordered Sets on each Lane. See § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding .
 - If eight consecutive TS1 Ordered Sets were received, speed_change bit is equal to 0b
 - If eight consecutive TS0 Ordered Sets were received, and the latest transition to Recovery.RcvrLock was from Recovery.Speed substate
 - If eight consecutive TS1 Ordered Sets were received, with the EC bits not equal to 00b

IMPLEMENTATION NOTE: REDOING EQUALIZATION §

A Downstream Port may use this provision to redo some parts of the Transmitter Equalization process using software help or some other implementation specific means while ensuring no transactions are in flight on the Link to avoid any timeouts.

- Next state for a Downstream Port is Recovery.Equalization if Recovery.RcvrLock was not entered from Configuration.Idle or Recovery.Idle and the Perform Equalization bit in the Link

Control 3 Register is set or an implementation specific mechanism determined equalization needs to be performed, following procedures described in § Section 4.2.4 .

The Downstream Port must ensure that no more than 2 TS1 Ordered Sets with EC=00b are transmitted due to being in Recovery.RcvrLock before transitioning to Recovery.Equalization and starting to transmit the TS0 / TS1 Ordered Sets.

- Transmitter sends TS1 Ordered Sets on all configured Lanes using the same Link and Lane numbers that were set after leaving Configuration . The speed_change bit (bit 7 of the Data Rate Identifier Symbol in TS1 Ordered Set) must be set to 1b if the directed_speed_change variable is set to 1b. The directed_speed_change variable is set to 1b if any configured Lane receives eight consecutive TS1 Ordered Sets with the speed_change bit set to 1b. Only those data rates greater than 2.5 GT/s should be advertised that can be supported reliably. In Non-Flit Mode of operation, the N_FTS value in the TS1 Ordered Set transmitted reflects the number at the current speed of operation. A device is allowed to change the supported data rates that it advertises when it enters this substate.

A Downstream Port that intends to redo equalization with a data rate change from 2.5 GT/s or 5.0 GT/s to 8.0 GT/s or 32.0 GT/s when Equalization Bypass to Highest NRZ Rate is supported must:

- Send EQ TS1 Ordered Sets with the speed_change bit set to 1b and advertising the following data rates:
 - 8.0 GT/s Data Rate Identifier if redo equalization is for 8.0 GT/s Data Rate
 - 32.0 GT/s Data Rate Identifier if redo equalization is for 32.0 GT/s Data Rate
- If the equalization redo attempt is initiated by the hardware as described in § Section 4.2.4 , then hardware must ensure that the Data Rate is 2.5 GT/s or 5.0 GT/s before initiating the attempt.
- If the equalization redo attempt is initiated by the software mechanism as described in § Section 4.2.4 , then software must ensure that the Data Rate is 2.5 GT/s or 5.0 GT/s before initiating the attempt.

A Downstream Port that intends to redo equalization with a data rate change from 8.0 GT/s to 16.0 GT/s, 16.0 GT/s to 32.0 GT/s, or 32.0 GT/s to 64.0 GT/s must:

- Send TS1 Ordered Sets with the Equalization Redo bit set to 1b, the speed_change bit set to 1b, and advertising the Data Rate Identifier at which equalization redo will be performed (16.0 GT/s, 32.0 GT/s, or 64.0 GT/s).
- If the equalization redo attempt is initiated by the hardware as described in § Section 4.2.4 , then hardware must ensure that the Data Rate is the following before initiating the attempt to redo equalization:
 - 8.0 GT/s if the equalization redo is for 16.0 GT/s Data Rate
 - 16.0 GT/s if the equalization redo is for 32.0 GT/s Data Rate
 - 32.0 GT/s if the equalization redo is for 64.0 GT/s Data Rate
- If the equalization redo attempt is initiated by the software mechanism as described in § Section 4.2.4 , then software must ensure that the Data Rate is the following before initiating the attempt to redo equalization:
 - 8.0 GT/s if the equalization redo is for 16.0 GT/s Data Rate
 - 16.0 GT/s if the equalization redo is for 32.0 GT/s Data Rate
 - 32.0 GT/s if the equalization redo is for 64.0 GT/s Data Rate

An Upstream Port must advertise the highest data rate support in the TS2 Ordered Sets it transmits in Recovery.RcvrCfg , and optionally in the TS1 Ordered Sets it transmits in this substate, unless the Upstream Port has determined that a problem unrelated to the highest data rate equalization prevents it from operating

reliably at the highest data rate at which equalization is being requested to be performed, if the eight consecutive Ordered Sets it receives are one of the following:

- EQ TS1 or EQ TS2 Ordered Sets with the speed_change bit set to 1b
- TS1 Ordered Sets with the Equalization Redo bit set to 1b or 128b/130b EQ TS2 Ordered Sets with the speed_change bit set to 1b.

Under other conditions, a device must not change the supported data rate values either in this substate or while in the Recovery.RcvrCfg or Recovery.Equalization substates. The ***successful_speed_negotiation*** variable is reset to 0b upon entry to this substate.

IMPLEMENTATION NOTE: HANDLING A REQUEST TO ADVERTISE 8.0 GT/S DATA RATE IDENTIFIER §

If an Upstream Port that is not advertising 8.0 GT/s Data Rate Identifiers receives EQ TSs with 8.0 GT/s Data Rate Identifiers and with the speed_change bit set in Recovery.RcvrLock , that indicates that the Downstream Port is attempting to switch the Link speed to 8.0 GT/s in order to perform the 8.0 GT/s Link Equalization Procedure. If for some reason the Upstream Port is unable or unwilling to switch to advertising 8.0 GT/s Data Rate Identifiers in the TS2 Ordered Sets it transmits once it transitions to Recovery.RcvrCfg , the 8.0 GT/s Link Equalization Procedure will not be performed in the current tenure in Recovery . This may cause the Downstream Port to permanently abandon its attempt to change the Link speed to 8.0 GT/s and perform the 8.0 GT/s Link Equalization Procedure, resulting in an operational link speed of less than 8.0 GT/s until after the link transitions through Detect and is re-trained. It is recommended that if an Upstream Port is for some temporary reason unable or unwilling to switch to advertising 8.0 GT/s Data Rate Identifiers in the condition described above, and does not intend to prohibit the Link from operating at 8.0 GT/s, that it perform one of the following two actions below as soon as is reasonable for it to do so:

- If the Upstream Port supports the Quiesce Guarantee mechanism for performing the Link Equalization Procedure, enter Recovery and advertise 8.0 GT/s Data Rate Identifiers with the speed_change bit set to 1b in the TSs that it sends. If Recovery.Equalization is not entered after changing speed to 8.0 GT/s and before entering Recovery.RcvrCfg at 8.0 GT/s (the Downstream Port did not direct an entry to Recovery.Equalization), it should set the Request Equalization and Quiesce Guarantee bits to 1b in the TS2 Ordered Sets sent at 8.0 GT/s in Recovery.RcvrCfg in order to request the Downstream Port to initiate the Link Equalization Procedure.
- Enter Recovery and advertise 8.0 GT/s Data Rate Identifiers with the speed_change bit cleared to 0b. The Downstream Port may then later initiate a speed change to 8.0 GT/s and perform the Link Equalization Procedure, though there is no guarantee that it will do so.

The process for handling a request to advertise 16.0 GT/s, 32.0 GT/s, or 64.0 GT/s Data Rate Identifier is similar to 8.0 GT/s Data Rate Identifier with 16.0 GT/s, 32.0 GT/s, or 64.0 GT/s Data Rate Identifier substituting 8.0 GT/s Data Rate Identifier and 128b/130b EQ TS2s substituting EQ TSs.

An Upstream Port must set the Selectable De-emphasis bit (bit 6 of Symbol 4) of the TS1 Ordered Sets it transmits to match the desired de-emphasis level at 5.0 GT/s. The mechanism an Upstream Port may adopt to request a de-emphasis level if it chooses to do so is implementation specific. It must also be noted that since the Upstream Port's request may not reach the Downstream Port due to bit errors in the TS1 Ordered Sets, the Upstream Port may attempt to re-request the desired de-emphasis level in subsequent entries to Recovery state when speed change is requested. If the Downstream Port intends to use the Upstream Port's de-emphasis information in Recovery.RcvrCfg , then it must record the value of the Selectable De-emphasis bit received in this state.

The Transmit Margin field of the Link Control 2 Register is sampled on entry to this substate and becomes effective on the transmit package pins within 192 ns of entry to this substate and remains effective until a new value is sampled on a subsequent entry to this substate from L0, L0s, or L1.

- After activating any inactive Lane, the Transmitter must wait for its TX common mode to settle before exiting Electrical Idle and transmitting the TS1 Ordered Sets with the following exceptions.
- When exiting from the L1.2 L1 PM Substate, common mode is permitted to be established passively during L1.0, and actively during Recovery. In order to ensure common mode has been established in Recovery.RcvrLock, the Downstream Port must maintain a timer, and the Downstream Port must not send TS2 Ordered Sets until a minimum of $T_{COMMONMODE}$ has elapsed since the Downstream Port has started transmitting TS1 Ordered Sets and has detected electrical idle exit on any Lane of the configured Link. See § Section 5.5.3.3.
- Implementations must note that the voltage levels may change after an early bit lock and Symbol or Block alignment since the new Transmit Margin field becomes effective within 192 ns after the other side enter Recovery.RcvrLock. The Receiver needs to reacquire bit lock and Symbol or Block alignment under those conditions.
 - a. Note: The directed_speed_change variable is set to 1b in L0 or L1 state for the side that is initiating a speed change. For the side that is not initiating a speed change, this bit is Set in this substate if the received TS Ordered Sets have the speed change bit Set. This bit is reset to 0b in the Recovery.Speed substate.
 - b. A device must accept all good TLPs and DLLPs it receives after entering this substate from L0 prior to receiving the first TS Ordered Set. If operating with 128b/130b encoding, any received TLPs and DLLPs are subject to the framing rules for 128b/130b encoding in § Section 4.2.2.3.
- Next state is Recovery.RcvrCfg if eight consecutive TS1 or TS2 Ordered Sets are received on all configured Lanes with the same Link and Lane numbers that match what is being transmitted on those same Lanes and the speed_change bit is equal to the directed_speed_change variable and the EC field is 00b in all the consecutive TS1 Ordered Sets if the current data rate is 8.0 GT/s or higher. See § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding.
 - If the Extended Synch bit is Set, the Transmitter must send a minimum of 1024 consecutive TS1 Ordered Sets before transitioning to Recovery.RcvrCfg.
 - If this substate was entered from Recovery.Equalization, the Upstream Port must evaluate the equalization coefficients or preset received by all Lanes that receive eight TS1 Ordered Sets and note whether they are different from the final set of coefficients or preset that was accepted in Phase 2 of the equalization process. Note: Mismatches are reported in Recovery.RcvrCfg by setting the Request Equalization bit of TS2 Ordered Sets.
- Otherwise, after a 24 ms timeout:
 - Next state is Recovery.RcvrCfg if the following two conditions are true:
 - Either of the following conditions are true:
 - Eight consecutive TS1 or TS2 Ordered Sets are received on any configured Lane with the same Link and Lane numbers that match what is being transmitted on the same Lane and the speed_change bit equal to 1b. See § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding.
 - The Link is operating in Flit Mode and eight consecutive TS2s are received on any configured Lane with the Link number set to PAD and the Lane number matches what is being transmitted on the same Lane. See § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding.
 - Either of the following conditions are true:
 - The current data rate of operation is greater than 2.5 GT/s.

- The highest data rate advertised in the Data Rate Identifier symbol of both the transmitted TS1 Ordered Sets and the (eight consecutive) received TS1 or TS2 Ordered Sets is 5.0 GT/s or greater.

IMPLEMENTATION NOTE: RECOMMENDED MECHANISM FOR LINK WIDTH REDUCTION IN FLIT MODE §

It is envisioned that when the Link Width is reduced in Flit Mode, both Ports will transition to Configuration from Recovery.RcvrCfg . If a Port proactively reduces the Link Width, it is recommended that ~~it waits~~ Port either: (1) Waits until it receives a TS2 Ordered Set on one of its Lanes or (2) Waits an additional 1 msec after the conditions are met for the Recovery.RcvrLock to Recovery.RcvrCfg transition. This decreases the likelihood that the initiator's Link Partner will take the timeout arc to Recovery.RcvrCfg since the initiator will continue to send TS1s with a valid Link and Lane number while in Recovery.RcvrLock .

- Else the next state is Recovery.Speed if the speed of operation has not changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 0b) and the current speed of operation is greater than 2.5 GT/s.
 - The new data rate to operate after leaving Recovery.Speed will be at 2.5 GT/s if 8b/10b or 128b/130b encoding is used. The new data rate of operation after leaving Recovery.Speed will be 32.0 GT/s if 1b/1b encoding is used. Note: This indicates that the Link was unable to operate at the current data rate and the Link will operate at the lower data rate of either 2.5 GT/s data rate or 32.0 GT/s.
- Else the next state is Recovery.Speed if the operating speed has been changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (changed_speed_recovery = 1b; i.e., the arc to this substate has been taken from Recovery.Speed). The new data rate to operate after leaving Recovery.Speed is reverted back to the speed it was when Recovery was entered from L0 or L1 .

Note: This indicates that the Link was unable to operate at the new negotiated data rate and will revert back to the old data rate with which it entered Recovery from L0 or L1 .

- Else the next state is Configuration and the directed_speed_change variable is reset to 0b if the following conditions are true:
 - If any of the configured Lanes that are receiving a TS1 or TS2 Ordered Set have received at least one TS1 or TS2 Ordered Set with Link and Lane numbers that match what is being transmitted on those same Lanes. See § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding .
 - The operating speed has not changed to a mutually negotiated data rate (i.e., changed_speed_recovery = 0b) since entering Recovery .

and at least one of the following conditions is true:

- The directed_speed_change variable is equal to 0b and the speed_change bit on the received TS1 or TS2 Ordered Set is equal to 0b.
 - The current data rate of operation is 2.5 GT/s and 2.5 GT/s data rate is the highest commonly advertised data rate among the transmitted TS1 Ordered Sets and the received TS1 or TS2 Ordered Set(s).
- Otherwise, the next state is Detect .

IMPLEMENTATION NOTE: EXAMPLE SHOWING SPEED CHANGE ALGORITHM BETWEEN 2.5 GT/S AND 5.0 GT/S §

Suppose a Link connects two greater than 5.0 GT/s capable components, A and B. The Link comes up to the L0 state in 2.5 GT/s data rate. Component A decides to change the speed to greater than 5.0 GT/s, sets the directed_speed_change variable to 1b and enters Recovery.RcvrLock from L0. Component A sends TS1 Ordered Sets with the speed_change bit set to 1b and advertises the 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s data rates. Component B sees the first TS1 in L0 state and enters Recovery.RcvrLock state. Initially, component B sends TS1s with the speed_change set to 0b. Component B will start sending the speed_change indication in its TS1 after it receives eight consecutive TS1 Ordered Sets from component A and advertises all of the data rates it can support. Component B will enter Recovery.RcvrCfg from where it will enter Recovery.Speed. Component A will wait for eight consecutive TS1/TS2 with speed_change bit set from component B before moving to Recovery.RcvrCfg and on to Recovery.Speed. Both component A and component B enter Recovery.Speed and record 8.0 GT/s as the maximum speed they can operate with. The directed_speed_change variable will be reset to 0b when in Recovery.Speed. When they enter Recovery.RcvrLock from Recovery.Speed, they will operate at 8.0 GT/s and send TS1s with speed_change set to 0b. If both sides work well at 8.0 GT/s, they will continue on to Recovery.RcvrCfg and enter L0 through Recovery.Idle at 8.0 GT/s. However, if component B fails to achieve Symbol lock, it will timeout in Recovery.RcvrLock and enters Recovery.Speed. Component A would have moved on to Recovery.RcvrCfg but would see the Electrical Idle after receiving TS1s at 8.0 GT/s after component B enters Recovery.Speed. This will cause component A to move to Recovery.Speed. After entering Recovery.Speed for the second time, both sides will revert back to the speed they operated with prior to entering the Recovery state (2.5 GT/s). Both sides will enter L0 from Recovery in 2.5 GT/s. Component A may initiate the directed_speed_change variable for a second time, requesting 8.0 GT/s data rate in its Data Rate Identifier, go through the same steps, fail to establish the 8.0 GT/s data rate and go back to L0 in 2.5 GT/s data rate. On the third attempt, however, component A may decide to only advertise 2.5 GT/s and 5.0 GT/s data rates and successfully establish the Link at 5.0 GT/s data rate and enter L0 at that speed. However, if either side entered Detect, that side should advertise all of the data rates it can support, since there may have been a hot plug event.

4.2.7.4.2 Recovery.Equalization §

If this state was entered from Recovery.RcvrLock, Transmitter sends either TS0 or TS1 Ordered Sets on all configured Lanes, as described in § Table 4-63, using the same Link and Lane numbers that were set after leaving Configuration. A Receiver must consider any TS1 or TS0 Ordered Set to be received only after it obtains Block Alignment in that Lane.

If this state was entered from Loopback.Entry :

- Transmitter sends either TS0 or TS1 Ordered Sets, as described in § Table 4-63, on all Lanes that detected a Receiver during Detect using the Link and Lane numbers defined in Loopback.Entry.
- The Lane under test is the only Lane that participates in the equalization procedure.
- The Lanes that are not under test must not be included in the equalization procedure and anything received by them is permitted to be ignored. The Lanes that are not under test must have their Transmitter preset values set to P4 / Q0. The sole purpose of the lanes that are not under test is to create the noise that is needed in Loopback.Active.

The Lanes must transmit the proper type of Ordered Set (TS0 vs TS1) and check for the receipt of the proper Ordered Set (TS0 vs TS1), according to § Table 4-63, in Recovery.Equalization anywhere TS0 / TS1 is mentioned.

Table 4-63 Use of TS0 or TS1 Ordered Sets in different phases §

| Current Data Rate / Port | Phase 0 / Phase 1 | Phase 2 | Phase 3 |
|--|-------------------|---------------------|---------------------|
| 8.0 GT/s, 16.0 GT/s, or 32.0 GT/s; Upstream/Downstream Lanes | TS1 | TS1 | TS1 |
| 64.0 GT/s Downstream Lanes | TS0 | TS0 followed by TS1 | TS1 |
| 64.0 GT/s Upstream Lanes | TS0 | TS0 | TS0 followed by TS1 |

IMPLEMENTATION NOTE: TS0 TO TS1 TRANSITIONS §

All the TS0 to TS1 transitions are expected and initiated by a Port so that its Receiver is prepared for the NRZ to PAM4 transition.

The first transition occurs for the Downstream Lanes when entering Phase 2. Prior to that, the Downstream Lanes send TS0 with EC=00b until they get their receiver to a target BER of 1E-4 and then they send EC = 01b. Next, they wait to receive EC = 10b from the Upstream Port to make the transition. When the Upstream Port sends EC = 10b, it is guaranteed of the transition within a fixed time since the Downstream Port has already acquired its target BER of 1E-4.

The other transition happens during Phase 3 for the Upstream Lanes, when the Upstream Port receives EC=11b. Since the Downstream Port sends EC=11b, it is expecting the NRZ to PAM4 transition.

We have also made a provision for the Retimer to request extended EQ during Phase 0 & 1 for 64.0 GT/s equalization.

4.2.7.4.2.1 Downstream Lanes §

Upon entry to this substate:

- Current phase is Phase 1
 - If the data rate of operation is 8.0 GT/s:
 - The Equalization 8.0 GT/s Phase 1 Successful , Equalization 8.0 GT/s Phase 2 Successful , Equalization 8.0 GT/s Phase 3 Successful , Link Equalization Request 8.0 GT/s , and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register and the Perform Equalization bit of the Link Control 3 Register are all set to 0b.
 - The equalization_done_8GT_data_rate variable is set to 1b.
 - If the data rate of operation is 16.0 GT/s:
 - The Equalization 16.0 GT/s Phase 1 Successful , Equalization 16.0 GT/s Phase 2 Successful , Equalization 16.0 GT/s Phase 3 Successful , Link Equalization Request 16.0 GT/s , and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register and the Perform Equalization bit of the Link Control 3 Register are all set to 0b.
 - The equalization_done_16GT_data_rate variable is set to 1b.
 - If the data rate of operation is 32.0 GT/s:

- The Equalization 32.0 GT/s Phase 1 Successful , Equalization 32.0 GT/s Phase 2 Successful , Equalization 32.0 GT/s Phase 3 Successful , Link Equalization Request 32.0 GT/s , and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register and the Perform Equalization bit of the Link Control 3 Register are all set to 0b.
- The equalization_done_32GT_data_rate variable is set to 1b.
- If the data rate of operation is 64.0 GT/s:
 - The Equalization 64.0 GT/s Phase 1 Successful , Equalization 64.0 GT/s Phase 2 Successful , Equalization 64.0 GT/s Phase 3 Successful , Link Equalization Request 64.0 GT/s , and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register and the Perform Equalization bit of the Link Control 3 Register are all set to 0b.
 - The equalization_done_64GT_data_rate variable is set to 1b.
- The start_equalization_w_preset variable is set to 0b.

4.2.7.4.2.1.1 Phase 1 of Transmitter Equalization §

- Transmitter sends TS0 / TS1 Ordered Sets using the Transmitter preset settings for the current data rate of operation. In the TS0 / TS1 Ordered Sets, the EC field is set to 01b. For TS0 ordered sets, the EC field is initially set to 00b. After two consecutive TS0 ordered sets are received with Retimer Equalization Extend bit set to 0b, the EC field is set to 01b. The TS0 / TS1 Transmitter Preset bits of each Lane are set to the value of its corresponding Transmitter preset setting for the current data rate. The FS and LF fields are set to the appropriate values. The Post-cursor Coefficient field is set to the value corresponding to the Lane's Transmitter Preset bits if TS1 Ordered Sets are transmitted. The Transmitter Preset settings, for each configured Lane, must be chosen as follows:
 1. If Recovery.Equalization was entered from Loopback.Entry :
 - If EQ TS1 Ordered Sets directed the device from Configuration.Linkwidth.Start to Loopback.Entry , the Transmitter preset value specified in the Preset field of the EQ TS1 Ordered Sets must be used by the Lane under test.
 - If standard TS1 Ordered Sets directed the device from Configuration.Linkwidth.Start to Loopback.Entry , an implementation specific method must be used to choose a supported Transmitter Preset value for use.
 - If perform_equalization_for_loopback_64GT is 1b, Loopback Follower must advertise 64.0 GT/s support in the transmitted TS0 / TS1 Ordered Sets (i.e., Data Rate Identifier must use the Flit Mode Encoding).
 2. Else, if eight consecutive 128b/130b EQ TS2 Ordered Sets were received with supported Transmitter preset values in the most recent transition through Recovery.RcvrCfg and the current data rate of operation is 16.0 GT/s or higher, the Transmitter preset value requested in the 128b/130b EQ TS2 Ordered Sets must be used.
 3. Else, if eight consecutive EQ TS2 Ordered Sets were received with supported Transmitter preset values in the most recent transition through Recovery.RcvrCfg , the current data rate of operation is 32.0 GT/s, and equalization bypass to 32.0 GT/s is being performed, the Transmitter preset value requested in the EQ TS2 Ordered Sets must be used.
 4. Else, if the Transmitter preset setting specified by the Downstream Port 8.0 GT/s Transmitter Preset field of the Lane Equalization Control Register Entry (for operation at the 8.0 GT/s data rate) or the Downstream Port 16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register Entry (for operation at the 16.0 GT/s data rate) or the Downstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry (for operation at the 32.0 GT/s data rate) or the Downstream Port 64.0 GT/s Transmitter Preset field of the 64.0 GT/s Lane Equalization Control Register

- Entry (for operation at the 64.0 GT/s data rate) is a supported value and is not a Reserved value, it must be used.
5. Else, use an implementation specific method to choose a supported Transmitter preset setting for use.
- The Downstream Port is permitted to wait for up to 500 ns after entering Phase 1 before evaluating received information for TS0 / TS1 Ordered Sets if it needs the time to stabilize its Receiver logic.
 - Next phase is Phase 2 if all configured Lanes receive two consecutive TS0 / TS1 Ordered Sets with EC=01b and the Downstream Port wants to execute Phase 2 and Phase 3. When the perform_equalization_for_loopback variable is 1b and the Downstream Port's Flit Mode Supported field of its PCI Express Capabilities Register is set to 1b, Phase 2 and Phase 3 must be executed.
 - The Receiver must complete its bit lock process and then recognize Ordered Sets within 2 ms after receiving the first bit of the first valid Ordered Set on its Receiver pin.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 1 Successful bit of the Link Status 2 Register is set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 1 Successful bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s and perform_equalization_for_loopback is 0b, the Equalization 32.0 GT/s Phase 1 Successful bit of the 32.0 GT/s Status Register is set to 1b.
 - If the data rate is 64.0 GT/s and perform_equalization_for_loopback is 0b, the Equalization 64.0 GT/s Phase 1 Successful bit of the 64.0 GT/s Status Register is set to 1b.
 - The LF and FS values received in the two consecutive TS0 / TS1 Ordered Sets must be stored for use during Phase 3, if the Downstream Port wants to adjust the Upstream Port's Transmitter coefficients.
 - Next state is Recovery.RcvrLock if all configured Lanes receive two consecutive TS0 / TS1 Ordered Sets with EC=01b, perform_equalization_for_loopback is 0b and the Downstream Port does not want to execute Phase 2 and Phase 3.
 - If the data rate is 8.0 GT/s, The Equalization 8.0 GT/s Phase 1 Successful, Equalization 8.0 GT/s Phase 2 Successful, Equalization 8.0 GT/s Phase 3 Successful, and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register are set to 1b.
 - If the data rate is 16.0 GT/s, The Equalization 16.0 GT/s Phase 1 Successful, Equalization 16.0 GT/s Phase 2 Successful, Equalization 16.0 GT/s Phase 3 Successful, and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register are set to 1b.
 - If the data rate is 32.0 GT/s, The Equalization 32.0 GT/s Phase 1 Successful, Equalization 32.0 GT/s Phase 2 Successful, Equalization 32.0 GT/s Phase 3 Successful, and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register are set to 1b.
 - If the data rate is 64.0 GT/s, The Equalization 64.0 GT/s Phase 1 Successful, Equalization 64.0 GT/s Phase 2 Successful, Equalization 64.0 GT/s Phase 3 Successful, and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register are set to 1b.
 - If the data rate is 64.0 GT/s, the Transmitter must send 24 TS0 Ordered Sets with EC=00b prior to transitioning to Recovery.RcvrLock
 - Note: A transition to Recovery.RcvrLock might be used in the case where the Downstream Port determines that Phase 2 and Phase 3 are not needed based on the platform and channel characteristics.
 - Next state is Loopback.Entry after a 24 ms timeout if perform_equalization_for_loopback is 1b.
 - Else, next state is Recovery.Speed after a 24 ms timeout if perform_equalization_for_loopback is 0b.
 - successful_speed_negotiation is set to 0b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 Register is set to 1b.

- If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
- If the data rate is 32.0 GT/s, the Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
- If the data rate is 64.0 GT/s, the Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

4.2.7.4.2.1.2 Phase 2 of Transmitter Equalization §

- The Transmitter sends TS0 Ordered Sets with EC=10b if the data rate is 64.0 GT/s and all Lanes have not received two consecutive TS0 Ordered Sets with EC=10b since entering this Phase; else it sends TS1 Ordered Sets.
- Transmitter sends TS1 Ordered Sets with EC = 10b and the coefficient settings, set on each Lane independently, as follows:
 - If two consecutive TS0 / TS1 Ordered Sets with EC=10b have been received since entering Phase 2, or two consecutive TS0 / TS1 Ordered Sets with EC=10b and a preset or set of coefficients (as specified by the Use Preset bit) different than the last two consecutive TS0 / TS1 Ordered Sets with EC=10b:
 - If the preset or coefficients requested in the most recent two consecutive TS0 / TS1 Ordered Sets are legal and supported (see § Section 4.2.4):
 - Change the transmitter settings to the requested preset or coefficients such that the new settings are effective at the Transmitter pins within 500 ns of when the end of the second TS0 / TS1 Ordered Set requesting the new setting was received at the Receiver pin. The change of Transmitter settings must not cause any illegal voltage level or parameter at the Transmitter pin for more than 1 ns.
 - In the transmitted TS1 Ordered Sets, the Transmitter Preset bits are set to the requested preset (for a preset request), the Pre-cursor, Cursor, and Post-cursor Coefficient fields are set to the Transmitter settings (for a preset or a coefficients request), and the Reject Coefficient Values bit is Clear.
 - Else (the requested preset or coefficients are illegal or unsupported): Do not change the Transmitter settings used, but reflect the requested preset or coefficient values in the transmitted TS1 Ordered Sets and set the Reject Coefficient Values bit to 1b.
 - Else: the preset and coefficients currently being used by the Transmitter.
 - Next phase is Phase 3 if all configured Lanes receive two consecutive TS0 / TS1 Ordered Sets with EC=11b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 2 Successful bit of the Link Status 2 Register is set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 2 Successful bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s and perform_equalization_for_loopback is 0b, the Equalization 32.0 GT/s Phase 2 Successful bit of the 32.0 GT/s Status Register is set to 1b.
 - If the data rate is 64.0 GT/s and perform_equalization_for_loopback is 0b, the Equalization 64.0 GT/s Phase 2 Successful bit of the 64.0 GT/s Status Register is set to 1b.
 - For data rates less than 64.0 GT/s, next state is Loopback.Entry after a 32 ms timeout with a tolerance of -0 ms and +4 ms if perform_equalization_for_loopback is 1b.
 - For the data rate of 64.0 GT/s, Next state is Loopback.Entry after a 64 ms timeout with a tolerance of -0 ms and +4 ms if perform_equalization_for_loopback is 1b.

- Else, if the data rate is less than 64.0 GT/s: next state is Recovery.Speed after a 32 ms timeout with a tolerance of -0 ms and +4 ms
 - successful_speed_negotiation is set to 0b.
 - If the data rate is 8.0 GT/s, The Equalization 8.0 GT/s Complete bit of the Link Status 2 Register is set to 1b.
 - If the data rate is 16.0 GT/s, The Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s, The Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
- Else, if the data rate is 64.0 GT/s: next state is Recovery.Speed after a 64 ms timeout with a tolerance of -0 ms and +4 ms
 - successful_speed_negotiation is set to 0b.
 - The Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

4.2.7.4.2.1.3 Phase 3 of Transmitter Equalization §

- Transmitter sends TS1 Ordered Sets with EC = 11b
- The Port must evaluate and arrive at the optimal settings independently on each Lane. When perform_equalization_for_loopback is 1b, the equalization procedure is only performed on the Lane under test. To evaluate a new preset or coefficient setting that is legal, as per the rules in § Section 4.2.4 and § Chapter 8.:
 - Request a new preset by setting the Transmitter Preset bits to the desired value and set the Use Preset bit to 1b. Alternatively, request a new set of coefficients by setting the Pre-cursor, Cursor, and Post-Cursor Coefficient fields to the desired values and set the Use Preset bit to 0b. Once a request is made, it must be continuously requested for at least 1 µs or until the evaluation of the request is completed, whichever is later.
 - Wait for the required time (500 ns plus the roundtrip delay including the logic delays through the Downstream Port) to ensure that, if accepted, the Upstream Port is transmitting using the requested settings. Obtain Block Alignment and then evaluate the incoming Ordered Sets. Note: The Downstream Port may simply ignore anything it receives during this waiting period as the incoming bit stream may be illegal during the transition to the requested settings. Hence the requirement to validate Block Alignment after this waiting period. If Block Alignment cannot be obtained after an implementation specific amount of time (in addition to the required waiting period specified above) it is recommended to proceed to perform Receiver evaluation on the incoming bit stream regardless.
 - If two consecutive TS1 Ordered Sets are received with the Transmitter Preset bits (for a preset request) or the Pre-cursor, Cursor, and Post-Cursor Coefficient fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Clear, then the requested setting was accepted and, depending on the results of Receiver evaluation, can be considered as a candidate final setting.
 - If two consecutive TS1 Ordered Sets are received with the Transmitter Preset bits (for a preset request) or the Pre-cursor, Cursor, and Post-Cursor Coefficient fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Set, then the requested setting was rejected and must not be considered as a candidate final setting.
 - If, after an implementation specific amount of time following the start of Receiver evaluation, no consecutive TS1s with the Transmitter Preset bits (for a preset request) or the Pre-Cursor, Cursor, and Post-Cursor Coefficient fields (for a coefficients request) identical to what was requested are received, then the requested setting must not be considered as a candidate final setting.

- The Downstream Port is responsible for setting the Reset EIEOS Interval Count bit in the TS1 Ordered Sets it transmits according to its evaluation criteria and requirements. The Use Preset bit of the received TS1 Ordered Sets must not be used to determine whether a request is accepted or rejected.

IMPLEMENTATION NOTE: RESET EIEOS AND COEFFICIENT/PRESET REQUESTS §

A Port may set Reset EIEOS Interval Count to 1b when it wants a longer PRBS pattern and subsequently clear it when it needs to obtain Block Alignment.

All TS1 Ordered Sets transmitted in this phase are requests. The first request maybe a new preset or a new coefficient request or a request to maintain the current link partner transmitter settings by reflecting the settings received in the two consecutive TS1 Ordered Sets with EC=11b that cause the transition to Phase 3.

- At 32.0 GT/s and below data rates, the total amount of time spent per preset or coefficients request from transmission of the request to the completion of the evaluation must be less than 2 ms. Implementations that need a longer evaluation time at the final stage of optimization may continue requesting the same preset or coefficient setting beyond the 2 ms limit but must adhere to the timeout (24 ms for 8.0, 16.0, and 32.0 GT/s and 48 ms for 64.0 GT/s) in this phase and must not take this exception more than two times. If the requester is unable to receive Ordered Sets within the timeout period, it may assume that the requested setting does not work in that Lane.
- At 64.0 GT/s and higher data rates, a device is permitted to evaluate each preset or coefficients request for an arbitrary amount of time. Evaluation must be carefully managed such that the search for an acceptable preset or coefficients can be successful. The total time spent in this Phase must still adhere to the timeout.
- All new preset or coefficient settings must be presented on all configured Lanes simultaneously. Any given Lane is permitted to continue to transmit the current preset or coefficients as its new value if it does not want to change the setting at that time.
- Next state is Loopback.Entry if the data rate of operation is 32.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied:
 - a. The Lane under test is operating at its optimal setting and it received two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
 - b. A 24 ms timeout with a tolerance of -0 ms and +2 ms.
- Next state is Loopback.Entry if the data rate of operation is 64.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied:
 - a. The Lane under test is operating at its optimal setting and all Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b are received.
 - b. A 48 ms timeout with a tolerance of -0 ms and +2 ms.
- Next state is Recovery.RcvrLock if perform_equalization_for_loopback is 0b, all configured Lanes are operating at their optimal setting and either the data rate of operation is 8.0 GT/s or all Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
 - If the data rate of operation is 8.0 GT/s: The Equalization 8.0 GT/s Phase 3 Successful and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register are set to 1b.

- If the data rate of operation is 16.0 GT/s: The Equalization 16.0 GT/s Phase 3 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register are set to 1b.
- If the data rate of operation is 32.0 GT/s: The Equalization 32.0 GT/s Phase 3 Successful and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register are set to 1b.
- If the data rate of operation is 64.0 GT/s: The Equalization 64.0 GT/s Phase 3 Successful and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register are set to 1b.
- Else, if the data rate is less than 64.0 GT/s: next state is Recovery.Speed after a timeout of 24 ms with a tolerance of -0 ms and +2 ms
 - successful_speed_negotiation is set to 0b.
 - If the data rate of operation is 8.0 GT/s: The Equalization 8.0 GT/s Complete bit of the Link Status 2 Register is set to 1b.
 - If the data rate of operation is 16.0 GT/s: The Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate of operation is 32.0 GT/s: The Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
- Else, if the data rate is 64.0 GT/s: next state is Recovery.Speed after a timeout of 48 ms with a tolerance of -0 ms and +2 ms
 - successful_speed_negotiation is set to 0b.
 - The Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

4.2.7.4.2.2 Upstream Lanes §

Upon entry to this substate:

- Current phase is Phase 0
 - If the data rate of operation is 8.0 GT/s:
 - The Equalization 8.0 GT/s Phase 1 Successful , Equalization 8.0 GT/s Phase 2 Successful , Equalization 8.0 GT/s Phase 3 Successful , Link Equalization Request 8.0 GT/s , and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register are all set to 0b
 - The equalization_done_8GT_data_rate variable is set to 1b.
 - If the data rate of operation is 16.0 GT/s:
 - The Equalization 16.0 GT/s Phase 1 Successful , Equalization 16.0 GT/s Phase 2 Successful , Equalization 16.0 GT/s Phase 3 Successful , Link Equalization Request 16.0 GT/s , and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register are all set to 0b.
 - The equalization_done_16GT_data_rate variable is set to 1b.
 - If the data rate of operation is 32.0 GT/s:
 - The Equalization 32.0 GT/s Phase 1 Successful , Equalization 32.0 GT/s Phase 2 Successful , Equalization 32.0 GT/s Phase 3 Successful , Link Equalization Request 32.0 GT/s , and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register are all set to 0b.
 - The equalization_done_32GT_data_rate variable is set to 1b.
 - If the data rate of operation is 64.0 GT/s:
 - The Equalization 64.0 GT/s Phase 1 Successful , Equalization 64.0 GT/s Phase 2 Successful , Equalization 64.0 GT/s Phase 3 Successful , Link Equalization Request 64.0 GT/s , and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register are all set to 0b.
 - The equalization_done_64GT_data_rate variable is set to 1b.

- The start_equalization_w_preset variable is set to 0b.

4.2.7.4.2.2.1 Phase 0 of Transmitter Equalization §

- If Recovery.Equalization was entered from Loopback.Entry, transmitter sends TS0 / TS1 Ordered Sets with the EC field set to 00b, the Transmitter Preset bits of the Lane is set to the value being used. The Pre-cursor Coefficient, Cursor Coefficient, and Post-cursor Coefficient fields are set to values corresponding to the Lane's Transmitter Preset bits if TS1 Ordered Sets are transmitted.
 - The Transmitter Preset settings for the Lane under test must be chosen as follows:
 - If EQ TS1 Ordered Sets directed the device from Configuration.Linkwidth.Start to Loopback.Entry, the Transmitter preset value specified in the Preset field of the EQ TS1 Ordered Sets must be used.
 - If standard TS1 Ordered Sets directed the device from Configuration.Linkwidth.Start to Loopback.Entry, an implementation specific method must be used to choose a supported Transmitter preset value for use.
 - If TS0 Ordered Sets are transmitted, while not recommended, components are permitted to use a supported Transmitter Preset value chosen by an ~~↓implementation-specific↑~~ ↓implementation specific↑ method.
- If the current data rate of operation is 8.0 GT/s, transmitter sends TS1 Ordered Sets using the Transmitter settings specified by the Transmitter Preset bits received in the EQ TS2 Ordered Sets during the most recent transition to 8.0 GT/s data rate from 2.5 GT/s or 5.0 GT/s data rate.

If the current data rate of operation is 16.0 GT/s, transmitter sends TS1 Ordered Sets using the 16.0 GT/s Transmitter settings specified by the Transmitter Preset bits received in the 128b/130b EQ TS2 Ordered Sets during the most recent transition to 16.0 GT/s data rate from 8.0 GT/s data rate.

If the current data rate of operation is 32.0 GT/s and perform_equalization_for_loopback is 0b, transmitter sends TS1 Ordered Sets using the 32.0 GT/s Transmitter settings specified by the Transmitter Preset bits received in the appropriate TS2 Ordered Sets during the most recent transition to the 32.0 GT/s data rate (EQ TS2 if equalization bypass was negotiated, 128b/130b EQ TS2 Ordered Sets if the most recent transition to the 32.0 GT/s data rate was from the 16.0 GT/s data rate).

If the current data rate of operation is 64.0 GT/s and perform_equalization_for_loopback is 0b, transmitter sends TS0 Ordered Sets using the 64.0 GT/s Transmitter settings specified by the Transmitter Preset bits received in the 128b/130b EQ TS2 Ordered Sets at 32.0 GT/s during the most recent transition to the 64.0 GT/s data rate.

Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use. Any reference to Transmitter Preset bits received in EQ TS2 Ordered Sets or 16.0 GT/s or higher data rate Transmitter Preset bits in 128b/130b EQ TS2 Ordered Sets (depending on the Data Rate) for the remainder of the Recovery.Equalization state is in reference to the presets determined above. In the TS1 Ordered Sets, the EC field is set to 00b, the Transmitter Preset bits of each Lane is set to the value it received in the Transmitter Preset bits of EQ TS2 Ordered Sets or 16.0 GT/s or higher data rate Transmitter Preset bits of 128b/130b EQ TS2 Ordered Sets, and the Pre-cursor Coefficient, Cursor Coefficient, and Post-cursor Coefficient fields are set to values corresponding to the Transmitter Preset bits.

- For Lanes that received a Reserved or unsupported Transmitter preset value in the EQ TS2 Ordered Sets or 128b/130b EQ TS2 Ordered Sets (depending on the Data Rate): in the TS1 / TS0 Ordered Sets, the Transmitter Preset field is set to the received Transmitter preset value, the Reject Coefficient Values bit is Set (applies to TS1 Ordered Sets only) and the Coefficient fields are set to values corresponding to the implementation specific Transmitter preset setting chosen by the Lane.⁹⁴

⁹⁴. An earlier version of this specification permitted the Reject Coefficient Values bit to be clear for this case. This is not recommended, but is permitted.

- For Lanes that did not receive EQ TS2 Ordered Sets or 128b/130b EQ TS2 Ordered Sets (depending on the Data Rate): in the TS1 / TS0 Ordered Sets, the Transmitter Preset field is set to the implementation specific Transmitter preset value chosen by the Lane, the Reject Coefficient Values bit is Clear (applies to TS1 Ordered Sets only), and the Coefficient fields are set to values corresponding to the same implementation specific Transmitter preset value chosen by the Lane and advertised in the Transmitter Preset bits.⁹⁵

IMPLEMENTATION NOTE: REJECT COEFFICIENT VALUES WITH TS0 ORDERED SETS §

The Reject Coefficient Values bit is intentionally omitted from the TS0 Ordered Set definition. As a result, Upstream Lanes are not able to explicitly indicate that a Reserved or unsupported Transmitter preset value was received in the 128b/130b EQ TS2 Ordered Sets at 32.0 GT/s during the most recent transition to the 64.0 GT/s data rate. In order to determine whether an Upstream Lane is using said Transmitter Preset, the Downstream Lane is permitted to observe the Transmitter Preset setting in the TS0 Ordered Sets received in Recovery.Equalization Phase 1 with the EC field set to 00b or 01b.

- The Upstream Port is permitted to wait for up to 500 ns after entering Phase 0 before evaluating receiver information for TS0 / TS1 Ordered Sets if it needs the time to stabilize its Receiver logic.
- Next phase is Phase 1 if all the configured Lanes receive two consecutive TS1 Ordered Sets with EC = 01b or if all the configured Lanes receive two consecutive TS0 Ordered Sets with EC = 01b and Retimer Equalization Extend bit set to 0b
 - The Receiver must complete its bit lock process and then recognize Ordered Sets within 2 ms after receiving the first bit of the first valid Ordered Set on its Receiver pin.
 - The LF and FS values received in the two consecutive TS0 / TS1 Ordered Sets must be stored for use during Phase 2 if the Upstream Port wants to adjust the Downstream Port's Transmitter coefficients.
- Next state is Loopback.Entry after a 12 ms timeout if perform_equalization_for_loopback is 1b.
- Else, next state is Recovery.Speed after a 12 ms timeout.
 - successful_speed_negotiation is set to 0b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 Register is set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s, the Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
 - If the data rate is 64.0 GT/s, the Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

95. An earlier version of this specification permitted the Transmitter Preset bits to be undefined and the Reject Coefficient Values bit to be clear for this case. This is not recommended, but is permitted

4.2.7.4.2.2.2 Phase 1 of Transmitter Equalization §

- Transmitter sends TS0 / TS1 Ordered Sets using the Transmitter settings determined in Phase 0. In the TS0 / TS1 Ordered Sets, the EC field is set to 01b, and the FS, LF, and (in TS1 Ordered Sets only) Post-cursor Coefficient fields of each Lane are set to values corresponding to the Lane's current Transmitter settings.
- If Recovery.Equalization was entered from Loopback.Entry and perform_equalization_for_loopback_64GT is 1b, Loopback Follower must advertise 64.0 GT/s support in the transmitted TS0 / TS1 Ordered Sets (i.e., Data Rate Identifier must use the Flit Mode Encoding).
- Next phase is Phase 2 if all configured Lanes receive two consecutive TS0 / TS1 Ordered Sets with EC=10b
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 1 Successful bit of the Link Status 2 Register are set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 1 Successful bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s and perform_equalization_for_loopback is 0b, the Equalization 32.0 GT/s Phase 1 Successful bit of the 32.0 GT/s Status Register is set to 1b.
 - If the data rate is 64.0 GT/s and perform_equalization_for_loopback is 0b, the Equalization 64.0 GT/s Phase 1 Successful bit of the 64.0 GT/s Status Register is set to 1b.
- Next state is Loopback.Entry after a 12 ms timeout if perform_equalization_for_loopback is 1b.
- Next state is Recovery.RcvrLock if all configured Lanes receive eight consecutive TS0 / TS1 Ordered Sets with EC=00b and perform_equalization_for_loopback is 0b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 1 Successful and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register are set to 1b
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 1 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register are set to 1b.
 - If the data rate is 32.0 GT/s, the Equalization 32.0 GT/s Phase 1 Successful and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register are set to 1b.
 - If the data rate is 64.0 GT/s, the Equalization 64.0 GT/s Phase 1 Successful and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register are set to 1b.
- Else, next state is Recovery.Speed after a 12 ms timeout if perform_equalization_for_loopback is 0b
 - successful_speed_negotiation is set to 0b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 Register for the current data rate of operation is set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s, the Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
 - If the data rate is 64.0 GT/s, the Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

4.2.7.4.2.2.3 Phase 2 of Transmitter Equalization §

- Transmitter sends TS0 / TS1 Ordered Sets with EC = 10b

- The Port must evaluate and arrive at the optimal settings independently on each Lane. When perform_equalization_for_loopback is 1b, the equalization procedure is only performed on the Lane under test. To evaluate a new preset or coefficient setting that is legal, as per the rules in § Section 4.2.4 and § Chapter 8 :
 - Request a new preset by setting the Transmitter Preset bits to the desired value and set the Use Preset bit to 1b. Alternatively, request a new set of coefficients by setting the Pre-cursor, Cursor, and Post-cursor Coefficient fields to the desired values and set the Use Preset bit to 0b. Once a request is made, it must be continuously requested for at least 1 μ s or until the evaluation of the request is completed, whichever is later.
 - Wait for the required time (500 ns plus the roundtrip delay including the logic delays through the Upstream Port) to ensure that, if accepted, the Downstream Port is transmitting using the requested settings. Obtain Block Alignment and then evaluate the incoming Ordered Sets. Note: The Upstream Port may simply ignore anything it receives during this waiting period as the incoming bit stream may be illegal during the transition to the requested settings. Hence the requirement to validate Block Alignment after this waiting period. If Block Alignment cannot be obtained after an implementation specific amount of time (in addition to the required waiting period specified above) it is recommended to proceed to perform Receiver evaluation on the incoming bit stream regardless.
 - If two consecutive TS1 Ordered Sets are received with the Transmitter Preset bits (for a preset request) or the Pre-cursor, Cursor, and Post-Cursor Coefficient fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Clear, then the requested setting was accepted and, depending on the results of Receiver evaluation, can be considered as a candidate final setting.
 - If two consecutive TS1 Ordered Sets are received with the Transmitter Preset bits (for a preset request) or the Pre-Cursor, Cursor, and Post-Cursor Coefficient fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Set, then the requested setting was rejected and must not be considered as a candidate final setting.
 - If, after an implementation specific amount of time following the start of Receiver evaluation, no consecutive TS1s with the Transmitter Preset bits (for a preset request) or the Pre-Cursor, Cursor, and Post-Cursor Coefficient fields (for a coefficients request) identical to what was requested are received, then the requested setting must not be considered as a candidate final setting.
 - The Upstream Port is responsible for setting the Reset EIEOS Interval Count bit in the TS0 / TS1 Ordered Sets it transmits according to its evaluation criteria and requirements. The Use Preset bit of the received TS1 Ordered Sets must not be used to determine whether a request is accepted or rejected.

IMPLEMENTATION NOTE: RESET EIEOS AND COEFFICIENT/PRESET REQUESTS §

A Port may set Reset EIEOS Interval Count to 1b when it wants a longer PRBS pattern and subsequently clear it when it needs to obtain Block Alignment.

All TS0 / TS1 Ordered Sets transmitted in this phase are requests. The first request maybe a new preset or a new coefficient request or a request to maintain the current link partner transmitter settings by reflecting the settings received in the two consecutive TS1 Ordered Sets with EC=10b that cause the transition to Phase 2.

- At 32.0 GT/s and below data rates, the total amount of time spent per preset or coefficients request from transmission of the request to the completion of the evaluation must be less than 2 ms. Implementations that need a longer evaluation time at the final stage of optimization may continue requesting the same setting beyond the 2 ms limit but must adhere to the timeout in this phase (24 ms for 8.0, 16.0, and 32.0 GT/s and 48 ms for 64.0 GT/s) and must not take this exception more than two times. If the requester is unable to receive Ordered Sets within the timeout period, it may assume that the requested setting does not work in that Lane.
- At 64.0 GT/s and higher data rates, a device is permitted to evaluate each preset or coefficients request for an arbitrary amount of time. Evaluation must be carefully managed such that the search for an acceptable preset or coefficients can be successful. The total time spent in this Phase must still adhere to the timeout.
- All new preset or coefficient settings must be presented on all configured Lanes simultaneously. Any given Lane is permitted to continue to transmit the current preset or coefficients as its new value if it does not want to change the setting at that time.
- If perform_equalization_for_loopback is 1b and the Lane under test is operating at its optimal setting and two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b are received, next phase is Phase 3.
 - If perform_equalization_for_loopback is 0b and all configured Lanes are operating at their optimal settings and either the data rate of operation is 8.0 GT/s or all Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b, next phase is Phase 3.
 - If the data rate of operation is 8.0 GT/s: The Equalization 8.0 GT/s Phase 2 Successful bit of the Link Status 2 Register are set to 1b.
 - If the data rate of operation is 16.0 GT/s: The Equalization 16.0 GT/s Phase 2 Successful bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate of operation is 32.0 GT/s: The Equalization 32.0 GT/s Phase 2 Successful bit of the 32.0 GT/s Status Register is set to 1b.
 - If the data rate of operation is 64.0 GT/s: The Equalization 64.0 GT/s Phase 2 Successful bit of the 64.0 GT/s Status Register is set to 1b.
 - Next state is Loopback.Entry after a timeout of 24 ms with a tolerance of -0 ms and +2 ms if perform_equalization_for_loopback is 1b and current data rate is less than 64.0 GT/s.
 - Next state is Loopback.Entry after a timeout of 48 ms with a tolerance of -0 ms and +2 ms if perform_equalization_for_loopback is 1b and current data rate is 64.0 GT/s.
 - Else, if the current data rate is less than 64.0 GT/s: next state is Recovery.Speed after a timeout of 24 ms with a tolerance of -0 ms and +2 ms
 - successful_speed_negotiation is set to 0b.
 - If the data rate of operation is 8.0 GT/s: The Equalization 8.0 GT/s Complete bit of the Link Status 2 Register is set to 1b.
 - If the data rate of operation is 16.0 GT/s: The Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate of operation is 32.0 GT/s: The Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
 - Else, if the current data rate is 64.0 GT/s: next state is Recovery.Speed after a timeout of 48 ms with a tolerance of -0 ms and +2 ms
 - successful_speed_negotiation is set to 0b.
 - The Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

4.2.7.4.2.2.4 Phase 3 of Transmitter Equalization §

- The Transmitter sends TS0 Ordered Sets with EC=11b if the data rate is 64.0 GT/s and all Lanes have not received two consecutive TS1 Ordered Sets with EC=11b since entering this Phase; else it sends TS1 Ordered Sets.
- Transmitter sends TS0 / TS1 Ordered Sets with EC = 11b and the coefficient settings, set on each configured Lane independently, as follows:
 - If two consecutive TS1 Ordered Sets with EC=11b have been received since entering Phase 3, or two consecutive TS1 Ordered Sets with EC=11b and a preset or set of coefficients (as specified by the Use Preset bit) different than the last two consecutive TS1 Ordered Sets with EC=11b:
 - If the preset or coefficients requested in the most recent two consecutive TS Ordered Sets are legal and supported (see § Section 4.2.4 and § Chapter 8.):
 - Change the transmitter settings to the requested preset or coefficients such that the new settings are effective at the Transmitter pins within 500 ns of when the end of the second TS1 Ordered Set requesting the new setting was received at the Receiver pin. The change of Transmitter settings must not cause any illegal voltage level or parameter at the Transmitter pin for more than 1 ns.
 - In the transmitted TS1 Ordered Sets, the Transmitter Preset bits are set to the requested preset (for a preset request), the Pre-cursor, Cursor, and Post-cursor Coefficient fields are set to the Transmitter settings (for a preset or a coefficients request), and the Reject Coefficient Values bit is Clear.
 - Else (the requested preset or coefficients are illegal or unsupported): Do not change the Transmitter settings used, but reflect the requested preset or coefficient values in the transmitted TS1 Ordered Sets and set the Reject Coefficient Values bit to 1b.
 - Else: the preset and coefficients currently being used by the Transmitter.
 - The Transmitter preset value initially transmitted on entry to Phase 3 can be the Transmitter preset value transmitted in Phase 0 for the same Data Rate or the Transmitter preset setting currently being used by the Transmitter.
- Next state is Loopback.Entry if perform_equalization_for_loopback is 1b, the current data rate is less than 64.0 GT/s, and one of the following conditions is satisfied:
 - a. The Lane under test receives two consecutive TS1 Ordered Sets with EC=00b.
 - b. A timeout of 32 ms with a tolerance of -0 ms and +4 ms.
- Next state is Loopback.Entry if perform_equalization_for_loopback is 1b, the current data rate is equal to 64.0 GT/s, and one of the following conditions is satisfied:
 - a. The Lane under test receives two consecutive TS1 Ordered Sets with EC=00b.
 - b. A timeout of 64 ms with a tolerance of -0 ms and +4 ms.
- Next state is Recovery.RcvrLock if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=00b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 3 Successful and Equalization 8.0 GT/s Complete bits of the Link Status 2 Register are set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 3 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status Register are set to 1b.
 - If the data rate is 32.0 GT/s, the Equalization 32.0 GT/s Phase 3 Successful and Equalization 32.0 GT/s Complete bits of the 32.0 GT/s Status Register are set to 1b.
 - If the data rate is 64.0 GT/s, the Equalization 64.0 GT/s Phase 3 Successful and Equalization 64.0 GT/s Complete bits of the 64.0 GT/s Status Register are set to 1b.

- Else, if the current data rate is less than 64.0 GT/s: next state is Recovery.Speed after a timeout of 32 ms with a tolerance of -0 ms and +4 ms
 - successful_speed_negotiation is set to 0b.
 - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 Register is set to 1b.
 - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status Register is set to 1b.
 - If the data rate is 32.0 GT/s, the Equalization 32.0 GT/s Complete bit of the 32.0 GT/s Status Register is set to 1b.
- Else, if current data rate is 64.0 GT/s: next state is Recovery.Speed after a timeout of 64 ms with a tolerance of -0 ms and +4 ms
 - successful_speed_negotiation is set to 0b.
 - The Equalization 64.0 GT/s Complete bit of the 64.0 GT/s Status Register is set to 1b.

4.2.7.4.3 Recovery.Speed §

- The Transmitter enters Electrical Idle and stays there until the Receiver Lanes have entered Electrical Idle, and then additionally remains there for at least 800 ns on a successful speed negotiation (i.e., successful_speed_negotiation = 1b) or at least 6 μ s on an unsuccessful speed negotiation (i.e., successful_speed_negotiation = 0b), but stays there no longer than an additional 1 ms. The frequency of operation is permitted to be changed to the new data rate only after the Receiver Lanes have entered Electrical Idle. If the negotiated data rate is 5.0 GT/s, and if operating in full swing mode, -6 dB de-emphasis level must be selected for operation if the select_deemphasis variable is 0b and -3.5 dB de-emphasis level must be selected for operation if the select_deemphasis variable is 1b. Note that if the link is already operating at the highest data rate supported by both Ports, Recovery.Speed is executed but the data rate is not changed.

An EIOSQ must be sent prior to entering Electrical Idle.

The DC common mode voltage is not required to be within specification.

An Electrical Idle condition exists on the Lanes if an EIOS is received on any of the configured Lanes or Electrical Idle is detected/inferred as described in § Section 4.2.5.4.

- On entry to this substate following a successful speed negotiation (i.e., successful_speed_negotiation = 1b), an Electrical Idle condition may be inferred on the Receiver Lanes if a TS1 or TS2 Ordered Set has not been received in any configured Lane in a time interval specified in § Table 4-50. (This covers the case where the Link is operational and both sides have successfully received TS Ordered Sets. Hence, a lack of a TS1 or TS2 Ordered Set in the specified interval can be interpreted as entry to Electrical Idle.)
- Else on entry to this substate following an unsuccessful speed negotiation (i.e., successful_speed_negotiation = 0b) if an exit from Electrical Idle has not been detected at least once in any configured Lane in a time interval specified in § Table 4-50. (This covers the case where at least one side is having trouble receiving TS Ordered Sets that was transmitted by the other agent, and hence a lack of exit from Electrical Idle in a longer interval can be treated as equivalent to entry to Electrical Idle.)
- Next state is Recovery.RcvrLock after the Transmitter Lanes are no longer required to be in Electrical Idle as described in the condition above.
 - If this substate has been entered from Recovery.RcvrCfg following a successful speed change negotiation (i.e., successful_speed_negotiation = 1b), the new data rate is changed on all the

configured Lanes to the highest common data rate advertised by both sides of the Link. The changed_speed_recovery variable is set to 1b.

- Else if this substate is being entered for a second time since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 1b), the new data rate will be the data rate at which the LTSSM entered Recovery from L0 or L1 . The changed_speed_recovery variable will be reset to 0b.
- Else if the latest encoding used before entering Recovery.Speed was 1b/1b, the new data rate will be 32.0 GT/s. The changed_speed_recovery variable remains reset at 0b.
- Else the new data rate will be 2.5 GT/s. The changed_speed_recovery variable remains reset at 0b.

Note: This represents the case where the frequency of operation in L0 was greater than 2.5 GT/s and one side could not operate at that frequency and timed out in Recovery.RcvrLock the first time it entered that substate from L0 or L1 .

- Next state is Detect after a 48 ms timeout if the Link Speed at which Recovery.Speed was entered was < 64.0 GT/s.
 - Note: This transition is not possible under normal conditions.
- Next state is Detect after a 96 ms timeout or 48 ms timeout if the Link Speed at which Recovery.Speed was entered was ≥ 64.0 GT/s.
 - Note: The 96 ms timeout is strongly recommended because the 48 ms timeout could cause an unintended timeout to Detect at 64.0 GT/s. This timeout to Detect can occur due to the 64.0 GT/s Recovery.Equalization Phase 2 timeout (64 ms) being larger than the Phase 1 timeout (12 ms) + Recovery.Speed timeout (48 ms).
- The directed_speed_change variable will be reset to 0b. The new data rate must be reflected in the Current Link Speed field of the Link Status Register .
 - On a Link bandwidth change, if successful_speed_negotiation is set to 1b and the Autonomous Change bit (bit 6 of Symbol 4) in the eight consecutive TS2 Ordered Sets received while in Recovery.RcvrCfg is set to 1b or the speed change was initiated by the Downstream Port for autonomous reasons (non-reliability and not due to the setting of the Link Retrain bit), the Link Autonomous Bandwidth Status bit of the Link Status Register is set to 1b.
 - Else: on a Link bandwidth change, the Link Bandwidth Management Status bit of the Link Status Register is set to 1b.

4.2.7.4.4 Recovery.RcvrCfg §

In Non-Flit Mode, Transmitter sends TS2 Ordered Sets on all configured Lanes using the same Link and Lane numbers that were set after leaving Configuration . In Flit Mode, Transmitter sends TS2 Ordered Sets on all configured Lanes with the Link number field sets as follows: if the LTSSM is initiating a Link width change by transitioning to Configuration from this State and the Lane will be removed from the Link, then the Link number field is set to PAD; otherwise it is set as the Link number. In 1b/1b TS2 Ordered Sets, if an equalization request is being communicated, Symbols 1,9 represent Equalization Byte 0 (instead of Link Number), the Request Equalization bit is set to 1b and the other fields are set as described in § Section 4.2.4 and § Table 4-39 . The LTSSM must not initiate a width change if speed_change is set to 1b or the LTSSM went through Recovery.Equalization followed by Recovery.RcvrLock prior to entering this substate. The speed_change bit (bit 7 of data rate identifier Symbol in TS2 Ordered Set) must be set to 1b if the directed_speed_change variable is already set to 1b. The N_FTS value in the transmitted TS2 Ordered Sets should reflect the number at the current data rate for Non-Flit Mode of operation. In Flit Mode, the appropriate Training Control bit of the TS2 Ordered set must be set, if the Port intends to drive the Link to Hot Reset , Disabled , or Loopback state immediately after exiting this sub-state.

Training Control bit is not a bit field in 1b/1b.

The Downstream Port must transmit EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 6 bit 7 set to 1b) on each configured Lane with the Transmitter Preset and Receiver Preset Hint fields set to the values specified by the Upstream 8.0 GT/s Port Transmitter Preset and the Upstream 8.0 GT/s Port Receiver Preset Hint fields from the corresponding Lane Equalization Control Register Entry if all of the following conditions are satisfied:

- a. The Downstream Port advertised 8.0 GT/s data rate support in Recovery.RcvrLock , and 8.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_8GT_data_rate variable is 0b or if the Perform Equalization bit in the Link Control 3 Register is Set or if an implementation specific mechanism determined equalization needs to be performed, following procedures described in § Section 4.2.4
- c. The current data rate of operation is 2.5 GT/s or 5.0 GT/s

The Downstream Port must transmit EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 6 bit 7 set to 1b) on each configured Lane with the Transmitter Preset bits set to the values specified by the Upstream Port 32.0 GT/s Transmitter Preset bits from the corresponding 32.0 GT/s Lane Equalization Control Register Entry if all of the following conditions are satisfied:

- a. The Downstream Port advertised 32.0 GT/s data rate support in Recovery.RcvrLock , and 32.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_32GT_data_rate variable is 0b or if the Perform Equalization bit in the Link Control 3 Register is Set or if an implementation specific mechanism determined equalization needs to be performed, following procedures described in § Section 4.2.4
- c. The equalization_done_8GT_data_rate and equalization_done_16GT_data_rate variables are 1b each
- d. Equalization Bypass to Highest NRZ Rate was negotiated between the components during Configuration state
- e. The current data rate of operation is 2.5 GT/s or 5.0 GT/s

The Downstream Port must transmit 128b/130b EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 7 bit 7 set to 1b) on each configured Lane with the Transmitter Preset bits set to the values specified by the Upstream Port 16.0 GT/s Transmitter Preset bits from the corresponding 16.0 GT/s Lane Equalization Control Register Entry if all of the following conditions are satisfied:

- a. The Downstream Port advertised 16.0 GT/s data rate support in Recovery.RcvrLock , and 16.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_16GT_data_rate variable is 0b or if the Perform Equalization bit in the Link Control 3 Register is set or an implementation specific mechanism determined equalization needs to be performed, following procedures described in § Section 4.2.4
- c. The current data rate of operation is 8.0 GT/s

The Downstream Port must transmit 128b/130b EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 7 bit 7 set to 1b) on each configured Lane with the Transmitter Preset bits set to the values specified by the Upstream Port 32.0 GT/s Transmitter Preset bits from the corresponding 32.0 GT/s Lane Equalization Control Register Entry if all of the following conditions are satisfied:

- a. The Downstream Port advertised 32.0 GT/s data rate support in Recovery.RcvrLock, and 32.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_32GT_data_rate variable is 0b or the Perform Equalization bit in the Link Control 3 Register is set or an implementation specific mechanism determined equalization needs to be performed, following procedures described in § Section 4.2.4
- c. The current data rate of operation is 16.0 GT/s

The Downstream Port must transmit 128b/130b EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 7 bit 7 set to 1b) on each configured Lane with the Transmitter Preset bits set to the values specified by the Upstream Port 64.0 GT/s Transmitter Preset bits from the corresponding 64.0 GT/s Lane Equalization Control Register Entry if all of the following conditions are satisfied:

- a. The Downstream Port advertised 64.0 GT/s data rate support in Recovery.RcvrLock, and 64.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_64GT_data_rate variable is 0b or the Perform Equalization bit in the Link Control 3 Register is set or an implementation specific mechanism determined equalization needs to be performed, following procedures described in § Section 4.2.4
- c. The current data rate of operation is 32.0 GT/s

The Upstream Port is permitted to transmit 128b/130b EQ TS2 Ordered Sets with the 16.0 GT/s Transmitter Preset bits set to implementation specific values if all of the following conditions are satisfied:

- a. The Upstream Port advertised 16.0 GT/s data rate support in Recovery.RcvrLock, and 16.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates, or optionally, but strongly recommended, in the Recovery.RcvrLock substate by the Downstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_16GT_data_rate variable is 0b or if directed by an implementation specific mechanism, following procedures described in § Section 4.2.4
- c. The current data rate of operation is 8.0 GT/s

The Upstream Port that intends to bypass equalization to the highest data rate of 32.0 GT/s or higher must transmit 8b/10b EQ TS2 Ordered Sets with the 32.0 GT/s Transmitter Preset bits set to implementation specific values if all of the following conditions are satisfied:

- a. The equalization bypass to the highest NRZ rate was negotiated during the Configuration state
- b. Either the Upstream Port requires precoding, or the Upstream Port intends to provide the Downstream Port's starting 32.0 GT/s Transmitter Preset for equalization
- c. The Upstream Port advertised 32.0 GT/s data rate support in Recovery.RcvrLock, and 32.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Downstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- d. The equalization_done_32GT_data_rate variable is 0b or if directed by an implementation specific mechanism, following procedures described in § Section 4.2.4
- e. The current data rate of operation is 2.5 GT/s or 5.0 GT/s

The Upstream Port is permitted to transmit 128b/130b EQ TS2 Ordered Sets with the 32.0 GT/s Transmitter Preset bits set to implementation specific values if all of the following conditions are satisfied:

- a. The Upstream Port advertised 32.0 GT/s data rate support in Recovery.RcvrLock , and 32.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates, or optionally, but strongly recommended, in the Recovery.RcvrLock substate by the Downstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_32GT_data_rate variable is 0b or if directed
- c. The current data rate of operation is 16.0 GT/s

The Upstream Port is permitted to transmit 128b/130b EQ TS2 Ordered Sets with the 64.0 GT/s Transmitter Preset bits set to implementation specific values if all of the following conditions are satisfied:

- a. The Upstream Port advertised 64.0 GT/s data rate support in Recovery.RcvrLock , and 64.0 GT/s data rate support has been advertised in the Configuration.Complete , Recovery.RcvrLock or Recovery.RcvrCfg substates by the Downstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b
- b. The equalization_done_64GT_data_rate variable is 0b or if directed
- c. The current data rate of operation is 32.0 GT/s

IMPLEMENTATION NOTE: ISSUES WITH SOFTWARE INITIATED SPEED CHANGE §

In prior versions of the specification, the EQ TS2 Ordered Sets from the Upstream Port may not be registered by the Downstream Port in the case where hardware autonomous equalization was not adopted. The Downstream Port enters Recovery and advertises a higher data rate (16.0 GT/s and above) for the first time with the software initiated speed change; the Upstream Port did not observe the new speed in Configuration.Complete . So if it was acting on the new speed based on observation in Recovery.RcvrCfg (as was required in prior versions of this specification), it may decide to switch from TS2 to EQ TS2 Ordered Sets while in Recovery.RcvrCfg . The Downstream Port may have satisfied its requirements with 8 consecutive TS2 Ordered Sets and missed the Preset request (for 16.0 GT/s or higher data rate) or Precoding request (for 32.0 GT/s or higher data rate) in the subsequent EQ TS2 Ordered Sets.

The situation described above can be avoided if the Upstream Port decides to send EQ TS2 Ordered Sets based on the higher speed being advertised by the Downstream Port while it is in Recovery.RcvrLock or if software requests re-equalization at the same data rate through the configuration register in the Downstream Port.

When using 128b/130b or 1b/1b encoding, Upstream and Downstream Ports use the Request Equalization, Equalization Request Data Rate, and Quiesce Guarantee bits of their transmitted TS2 Ordered Sets to communicate equalization requests as described in § Section 4.2.4 . When not requesting equalization, the Request Equalization, Equalization Request Data Rate, and Quiesce Guarantee bits must be set to 0b.

The **start_equalization_w_preset** variable is reset to 0b upon entry to this substate.

- On entry to this substate, a Downstream Port must set the select_deemphasis variable equal to the Selectable De-emphasis field in the Link Control 2 Register or adopt some implementation specific mechanism to set the select_deemphasis variable, including using the value requested by the Upstream Port in the eight consecutive TS1 Ordered Sets it received. A Downstream Port advertising 5.0 GT/s data rate support must set the Selectable De-emphasis bit (Symbol 4 bit 6) of the TS2 Ordered Sets it transmits identical to the select_deemphasis

variable. An Upstream Port must set its Autonomous Change bit (Symbol 4 bit 6) to 1b in the TS2 Ordered Set if it intends to change the Link bandwidth for autonomous reasons.

- For devices that support Link width upconfigure, it is recommended that the Electrical Idle detection circuitry be activated in the set of currently inactive Lanes in this substate, the Recovery.Idle substate, and Configuration.Linkwidth.Start substates, if the directed_speed_change variable is reset to 0b. This is done so that during a Link upconfigure, the side that does not initiate the upconfiguration does not miss the first EIEOSQ sent by the initiator during the Configuration.Linkwidth.Start substate.
- Next state is Recovery.Speed if all of the following conditions are true:
 - One of the following conditions is satisfied:
 - i. Eight consecutive TS2 Ordered Sets are received on any configured Lane with identical data rate identifiers, identical values in Symbol 6, and the speed_change bit set to 1b and eight consecutive TS2 Ordered Sets are standard TS2 Ordered Sets if either 8b/10b or 128b/130b encoding is used
 - ii. Eight consecutive EQ TS2 or 128b/130b EQ TS2 Ordered Sets are received on all configured Lanes with identical data rate identifiers, identical value in Symbol 6, and the speed_change bit set to 1b
 - iii. Eight consecutive EQ TS2 or 128b/130b EQ TS2 Ordered Sets are received on any configured Lane with identical data rate identifiers, identical value in Symbol 6, and the speed_change bit set to 1b and 1 ms has expired since the receipt of the eight consecutive EQ Ordered Sets on any configured Lane
 - iv. Eight consecutive and identical TS2 Ordered Sets are received on any configured Lane with the speed_change bit set to 1b when 1b/1b encoding is used
 - Either the current data rate is greater than 2.5 GT/s or greater than 2.5 GT/s data rate identifiers are set both in the transmitted and the (eight consecutive) received TS2 Ordered Sets
 - For 8b/10b encoding, at least 32 TS2 Ordered Sets, without being interrupted by any intervening EIEOS, are transmitted with the speed_change bit set to 1b after receiving one TS2 Ordered Set with the speed_change bit set to 1b in the same configured Lane. For 128b/130b and 1b/1b encoding, at least 128 TS2 Ordered Sets are transmitted with the speed_change bit set to 1b after receiving one TS2 Ordered Set with the speed_change bit set to 1b in the same configured Lane.

The data rate(s) advertised on the received eight consecutive TS2 Ordered Sets with the speed_change bit set is noted as the data rate(s) that can be supported by the other Port. The Autonomous Change bit (Symbol 4 bit 6) in these received eight consecutive TS2 Ordered Sets is noted by the Downstream Port for possible logging in the Link Status Register in Recovery.Speed substate. Upstream Ports must register the Selectable De-emphasis bit (bit 6 of Symbol 4) advertised in these eight consecutive TS2 Ordered Sets in the select_deemphasis variable. The new speed to change to in Recovery.Speed is the highest data rate that can be supported by both Ports on the Link.

For an Upstream Port, if the current data rate of operation is 2.5 GT/s or 5.0 GT/s and these eight consecutive TS2 Ordered Sets are EQ TS2 Ordered Sets advertising 8.0 GT/s as the highest data rate supported, it must set the start_equalization_w_preset variable to 1b and update the Upstream Port 8.0 GT/s Transmitter Preset and Upstream Port 8.0 GT/s Receiver Hint fields of the Lane Equalization Control Register Entry with the values received in the eight consecutive EQ TS2 Ordered Sets for the corresponding Lane.

For an Upstream Port, if the current data rate of operation is 2.5 GT/s or 5.0 GT/s and these eight consecutive TS2 Ordered Sets are EQ TS2 Ordered Sets advertising 32.0 GT/s as the highest data rate supported and equalization bypass to the highest NRZ rate was negotiated between the components during the Configuration state, it must set the start_equalization_w_preset variable to 1b and update the Upstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry with the values received in the eight consecutive EQ TS2 Ordered Sets for the corresponding Lane.

For an Upstream Port, if the current data rate of operation is 8.0 GT/s, 16.0 GT/s support is advertised by both ends, and these eight consecutive TS2 Ordered Sets are 128b/130b EQ TS2 Ordered Sets, it must set the start_equalization_w_preset variable to 1b and update the Upstream Port 16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register Entry with the values received in the eight consecutive 128b/130b EQ TS2 Ordered Sets for the corresponding Lane.

For an Upstream Port, if the current data rate of operation is 16.0 GT/s, 32.0 GT/s support is advertised by both ends, and these eight consecutive TS2 Ordered Sets are 128b/130b EQ TS2 Ordered Sets, it must set the start_equalization_w_preset variable to 1b and update the Upstream Port 32.0 GT/s Transmitter Preset field of the 32.0 GT/s Lane Equalization Control Register Entry with the values received in the eight consecutive 128b/130b EQ TS2 Ordered Sets for the corresponding Lane.

For an Upstream Port, if the current data rate of operation is 32.0 GT/s, 64.0 GT/s support is advertised by both ends, and these eight consecutive TS2 Ordered Sets are 128b/130b EQ TS2 Ordered Sets, it must set the start_equalization_w_preset variable to 1b and update the Upstream Port 64.0 GT/s Transmitter Preset field of the 64.0 GT/s Lane Equalization Control Register Entry with the values received in the eight consecutive 128b/130b EQ TS2 Ordered Sets for the corresponding Lane.

Any configured Lanes which do not receive EQ TS2 or 128b/130b EQ TS2 Ordered Sets meeting this criteria will use implementation dependent preset values when first operating at 8.0, 16.0, 32.0, or 64.0 GT/s prior to performing link equalization. A Downstream Port must set the start_equalization_w_preset variable to 1b if any of the following are true:

- the equalization_done_8GT_data_rate variable is 0b
- 16.0 GT/s support is advertised by both ends and the equalization_done_16GT_data_rate variable is 0b
- 32.0 GT/s support is advertised by both ends and the equalization_done_32GT_data_rate variable is 0b
- 64.0 GT/s support is advertised by both ends and the equalization_done_64GT_data_rate variable is 0b
- the Perform Equalization bit in Link Control 3 Register is Set
- an implementation specific mechanism determined that equalization needs to be performed, following procedures described in § Section 4.2.4.

A Downstream Port must record the 16.0 GT/s, 32.0 GT/s, or 64.0 GT/s Transmitter Preset settings advertised in the eight consecutive TS2 Ordered Sets received if they are 128b/130b EQ TS2 Ordered Sets, and 16.0 GT/s, 32.0 GT/s, or 64.0 GT/s support is advertised by both ends. The variable successful_speed_negotiation is set to 1b. Note that if the Link is already operating at the highest data rate supported by both Ports, Recovery.Speed is executed but the data rate is not changed. If 128b/130b or 1b/1b encoding is used and the Request Equalization bit is Set in the eight consecutive TS2 Ordered Sets, the Port must handle it as an equalization request as described in § Section 4.2.4.

- Next state is Recovery.Idle if the following two conditions are both true:
 - Eight consecutive TS2 Ordered Sets are received on all configured Lanes with the same Link and Lane number that match what is being transmitted on those same Lanes (see § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding) with identical data rate identifiers within each Lane and one of the following two sub-conditions are true:
 - the speed_change bit is 0b in the received eight consecutive TS2 Ordered Sets
 - current data rate is 2.5 GT/s and either no 5.0 GT/s, or higher, data rate identifiers are set in the received eight consecutive TS2 Ordered Sets, or no 5.0 GT/s, or higher, data rate identifiers are being transmitted in the TS2 Ordered Sets

- 16 TS2 Ordered Sets are sent after receiving one TS2 Ordered Set without being interrupted by any intervening EIEOS. The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b on entry to Recovery.Idle.
- In Flit Mode, if the received eight consecutive TS2 Ordered Sets have Hot_Reset_Request, Disable_Link_Request, or Loopback_Request asserted on any configured Lane, the corresponding directed bit must be set.

IMPLEMENTATION NOTE:

The above requirement ensures that both the components will immediately exit from Recovery.Idle to the corresponding Hot Reset / Disable / Loopback state without the follower having to send a data stream and waiting till the next scheduled SKP Ordered Set boundary.

- If the N_FTS value was changed, the new value must be used for future L0s states.
- When using 8b/10b encoding, Lane-to-Lane de-skew must be completed before leaving Recovery.RcvrCfg.
- The device must note the data rate identifier advertised on any configured Lane in the eight consecutive TS2 Ordered Sets described in this state transition. This will override any previously recorded value.
- When using 128b/130b or 1b/1b encoding and if the Request Equalization bit is Set in the eight consecutive TS2 Ordered Sets, the device must note it and follow the rules in § Section 4.2.4.
- Next state is Configuration in Non-Flit Mode if eight consecutive TS1 Ordered Sets are received on any configured Lanes with Link or Lane numbers that do not match what is being transmitted on those same Lanes and 16 TS2 Ordered Sets are sent after receiving one TS1 Ordered Set, either 8b/10b or 128b/130b encoding is used, and one of the following two conditions apply:
 - the speed_change bit is 0b on the received TS1 Ordered Sets
 - current data rate is 2.5 GT/s and either no 5.0 GT/s, or higher, data rate identifiers are set in the received eight consecutive TS1 Ordered Sets, or no 5.0 GT/s, or higher, data rate identifiers are being transmitted in the TS2 Ordered Sets

The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b if the LTSSM transitions to Configuration .

- If the N_FTS value was changed, the new value must be used for future L0s states.
- Next state is Configuration in Flit Mode if all the following conditions are true:
 - 1 msec has elapsed after receiving one TS1 or TS2 Ordered Set in any configured Lane in this sub-state.
 - One of the following conditions is true:
 - Eight consecutive TS1 Ordered Sets are received after receiving a TS2 Ordered Set on any configured Lane
 - Eight consecutive TS1 Ordered Sets are received with the Link or Lane number set to PAD and no TS2 Ordered Set was received on any configured Lane
 - Eight consecutive TS2 Ordered Sets are received with Link number PAD on any configured Lane or a PAD is being transmitted in the TS2 Ordered Sets in any configured Lane (see § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding)
 - At least 16 TS2 Ordered Sets have been transmitted after receiving either of the following:
 - One TS1 Ordered Set with Link or Lane number set to PAD

- One TS2 Ordered Set
- The speed_change bit is 0b on the received TS1 or TS2 Ordered Sets.

The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b if the LTSSM transitions to Configuration .

IMPLEMENTATION NOTE: REDUCING LINK WIDTH DUE TO ERRORS IN FLIT MODE §

With Flit Mode, since there is no upconfig support, the reason to make the transition for Recovery → Configuration is to reduce the Link width due to errors. In that case, the Port that wants to reduce the Link width should send PAD in the Link Number field on those Lanes it does not want to be part of the configured Link.

- Next state is Recovery.Speed if the speed of operation has changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 1b) and an EIOS has been detected or an Electrical Idle condition has been inferred/detected on any of the configured Lanes and no configured Lane received a TS2 Ordered Set since entering this substate (Recovery.RcvrCfg). The new data rate to operate after leaving Recovery.Speed will be reverted back to the speed of operation during entry to Recovery from L0 or L1 .

As described in § Section 4.2.5.4 , an Electrical Idle condition may be inferred if a TS1 or TS2 Ordered Set has not been received in a time interval specified in § Table 4-50 .

- Next state is Recovery.Speed if the speed of operation has not changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 0b) and the current speed of operation is greater than 2.5 GT/s and an EIOS has been detected or an Electrical Idle condition has been detected/inferred on any of the configured Lanes and no configured Lane received a TS2 Ordered Set since entering this substate (Recovery.RcvrCfg). The new data rate to operate after leaving Recovery.Speed will be 2.5 GT/s if 8b/10b or 128b/130b encoding is used. The new data rate to operate after leaving Recovery.Speed will be 32.0 GT/s if 1b/1b encoding is used.

As described in § Section 4.2.5.4 , an Electrical Idle condition may be inferred if a TS1 or TS2 Ordered Set has not been received in a time interval specified in § Table 4-50 .

Note: This transition implies that the other side was unable to achieve Symbol lock or Block alignment at the speed with which it was operating. Hence both sides will go back to the 2.5 GT/s speed of operation and neither device will attempt to change the speed again without exiting Recovery state. It should also be noted that even though a speed change is involved here, the changed_speed_recovery will be 0b.

- After a 48 ms timeout;
 - The next state is Detect if the current data rate is 2.5 GT/s or 5.0 GT/s.
 - The next state is Recovery.Idle if the idle_to_rlock_transitioned variable is less than FFh and the current data rate is 8.0 GT/s or higher.
 - i. The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b on entry to Recovery.Idle .
 - Else the next state is Detect .

4.2.7.4.5 Recovery.Idle §

- Next state is Disabled if directed.

- The clause “if directed” applies to a Downstream or optional crosslink Port that is instructed by a higher Layer to assert Disable_Link_Request in the Training Sets it transmits on the Link.

The clause, “if directed”, also applies to an Upstream Port in Flit Mode that transitioned to this state with Disable_Link_Request set in the Training Control field in any configured Lane in the eight consecutive TS2 Ordered Sets that resulted in the transition to this state.

- Next state is Hot Reset if directed.

- The clause “if directed” applies to a Downstream or optional crosslink Port that is instructed by a higher Layer to assert Hot_Reset_Request in the Training Sets it transmits on the Link.

The clause, “if directed”, also applies to an Upstream or optional crosslink Port in Flit Mode that transitioned to this state with Hot_Reset_Request asserted in the Training Control field in any configured Lane in the eight consecutive TS2 Ordered Sets that resulted in the transition to this state.

- Next state is Configuration if directed.

- The clause “if directed” applies to a Port that is instructed by a higher Layer to optionally re-configure the Link (i.e., different width Link).

- Next state is Loopback if directed to this state, and the Transmitter is capable of being a Loopback Lead, which is determined by implementation specific means.

- The clause “if directed” applies to a Port that is instructed by a higher Layer to assert Loopback_Request in the Training Sets it transmits on the Link, and the Port becomes the Loopback Lead.

- Next state is Disabled immediately after any configured Lane has Disable_Link_Request asserted in two consecutively received TS1 Ordered Sets.

- This behavior is only applicable to Upstream and optional crosslink Ports in Non-Flit Mode.

- Next state is Hot Reset immediately after any configured Lane has Hot_Reset_Request asserted in two consecutively received TS1 Ordered Sets.

- This behavior is only applicable to Upstream and optional crosslink Ports in Non-Flit Mode.

- Next state is Configuration if two consecutive TS1 Ordered Sets are received on any configured Lane with a Lane number set to PAD. See § Section 4.2.7 for more on Matching Link and Lane Numbers for 1b/1b Encoding.

- Note: A Port that optionally transitions to Configuration to change the Link configuration is guaranteed to send Lane numbers set to PAD on all Lanes.

- Note: It is recommended that in Non-Flit Mode, the LTSSM initiate a Link width up/downsizing using this transition to reduce the time it takes to change the Link width. In Flit Mode, the Recovery.RcvrCfg to Configuration transition should be used.

- Next state is Loopback if one of the following conditions is true:

- The Port is operating in Non-Flit Mode and any configured Lane has Loopback_Request asserted in two consecutive TS1 Ordered Sets.

- The port is operating in Flit Mode and the received eight consecutive TS2 Ordered Sets that resulted in the transition to this state had Loopback_Request asserted in the Training Control field in any configured Lane.

- The device receiving the Ordered Set with Loopback_Request asserted becomes the Loopback Follower.

- When using 8b/10b encoding, the Transmitter sends Idle data Symbols on all configured Lanes in Non-Flit Mode and IDLE Flits across all configured Lanes in Flit Mode.

- When using 128b/130b encoding in Non-Flit Mode:

- If the data rate is 8.0 GT/s, the Transmitter sends one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
- If the data rate is 16.0 GT/s or higher, the Transmitter sends one Control SKP Ordered Set followed immediately by one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
- If directed to other states, Idle Symbols do not have to be sent, and must not be sent with 128b/130b encoding, before transitioning to the other states (i.e., Disabled, Hot Reset, Configuration, or Loopback)

IMPLEMENTATION NOTE: EDS USAGE §

In 128b/130b encoding in Non-Flit Mode, on transition to Configuration or Loopback or Hot Reset or Disabled, an EDS must be sent if a Data Stream is active (i.e., an SDS Ordered Set has been sent). It is possible that the side that is not initiating Link Upconfigure has already transmitted SDS and transmitting Data Stream (Logical IDL) when it receives the TS1 Ordered Sets. In that situation, it will send EDS in the set of Lanes that are active before sending the TS1 Ordered Sets in Configuration .

- When using 1b/1b encoding or 128b/130b encoding in Flit Mode:
 - Transmitter sends one SDS Ordered Set sequence followed by a Control SKP Ordered Set on all configured Lanes followed by IDLE Flits to start a Data Stream.
 - If directed to other states, Idle Flits do not have to be sent, and must not be sent before transitioning to the other states (i.e., Disabled, Hot Reset, Configuration, or Loopback)
- When using 8b/10b encoding in Non-Flit Mode, next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes and 16 Idle data Symbols are sent after receiving one Idle data Symbol.
 - If software has written a 1b to the Retrain Link bit in the Link Control Register since the last transition to L0 from Recovery or Configuration , the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status Register to 1b.
- When using 128b/130b encoding in Non-Flit Mode, next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes, 16 Idle data Symbols are sent after receiving one Idle data Symbol, and this state was not entered by a timeout from Recovery.RcvrCfg
 - The Idle data Symbols must be received in Data Blocks.
 - Lane-to-Lane de-skew must be completed before Data Stream processing starts.
 - If software has written a 1b to the Retrain Link bit in the Link Control Register since the last transition to L0 from Recovery or Configuration , the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status Register to 1b.
 - The idle_to_rlock_transitioned variable is reset to 00h on transition to L0 .
- In Flit Mode, the next state is L0 if two consecutive IDLE Flits are received and the minimum number of IDLE Flits are sent after receiving one IDLE Flit and this state was not entered by a timeout from Recovery.RcvrCfg . The minimum number of IDLE Flits to send is 4 with 8b/10b or 128b/130b encoding and 8 with 1b/1b encoding.
 - Lane-to-Lane de-skew must be completed before Data Stream processing starts.
 - If software has written a 1b to the Retrain Link bit in the Link Control Register since the last transition to L0 from Recovery or Configuration , the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status Register to 1b.

- The idle_to_rlock_transited variable is reset to 00h on transition to L0
- Otherwise, after a 2 ms timeout:
 - If the idle_to_rlock_transited variable is less than FFh, the next state is Recovery.RcvrLock.
 - If the data rate is 8.0 GT/s or higher, the idle_to_rlock_transited variable is incremented by 1b upon transitioning to Recovery.RcvrLock.
 - If the data rate is 5.0 GT/s (or, if supported in 2.5 GT/s), the idle_to_rlock_transited variable is set to FFh, upon transitioning to Recovery.RcvrLock.
 - Else the next state is Detect

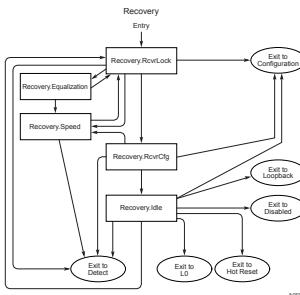


Figure 4-71 Recovery Substate Machine §

4.2.7.5 L0 §

This is the normal operational state. It includes the L0p state, where some Lanes can be in idle state.

- LinkUp = 1b (status is set true).
 - On receipt of an STP or SDP Symbol, the idle_to_rlock_transited variable is reset to 00h.
- For an Upstream Port, the directed_speed_change variable must not be set to 1b if it has never recorded greater than 2.5 GT/s data rate support advertised in Configuration.Complete or Recovery.RcvrCfg substates by the Downstream Port since exiting the Detect state.
- For a Downstream Port, the directed_speed_change variable must not be set to 1b if it has never recorded greater than 2.5 GT/s data rate support advertised in Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state. If greater than 2.5 GT/s data rate support has been noted, the Downstream Port must set the directed_speed_change variable to 1b if the Retrain Link bit of the Link Control Register is set to 1b and the Target Link Speed field in the Link Control 2 Register is not equal to the current Link speed.
- A Port supporting greater than 2.5 GT/s data rates must participate in the speed change even if the Link is not in DL_Active state if it is requested by the other side through the TS Ordered Sets.
- Next state is Recovery if directed to change speed (directed_speed_change variable = 1b) by a higher layer and any of the following three conditions are satisfied:
 - both sides support greater than 2.5 GT/s data rates and the Link is in DL_Active state
 - both sides support 8.0 GT/s or higher data rates, in order to perform Transmitter Equalization at a data rate supported by both sides, in which case the changed_speed_recovery bit is reset to 0b
 - an alternate protocol was selected by the Downstream Port and the current data rate of operation is not an operational data rate in the negotiated alternate protocol
- Next state is Recovery if directed to change Link width.

- The upper layer must not direct a Port to increase the Link width if the other Port did not advertise the capability to upconfigure the Link width during the Configuration state or if the Link is currently operating at the maximum possible width it negotiated on initial entry to the L0 state.
- Normally, the upper layer will not reduce width if upconfigure_capable is reset to 0b other than for reliability reasons, since the Link will not be able to go back to the original width if upconfigure_capable is 0b. A Port must not initiate reducing the Link width for reasons other than reliability if the Hardware Autonomous Width Disable bit in the Link Control Register is set to 1b.
- The decision to initiate an increase or decrease in the Link width, as allowed by the specification, is implementation specific.
- Next state is Recovery if a TS1 or TS2 Ordered Set is received on any configured Lane or an EIEOS is received on any configured Lane in 128b/130b or 1b/1b encoding.
- Next state is Recovery if directed to this state. If Electrical Idle is detected/inferred on all Lanes without receiving an EIOS on any Lane, the Port may transition to the Recovery state or may remain in L0. In the event that the Port is in L0 and the Electrical Idle condition occurs without receiving an EIOS, errors may occur and the Port may be directed to transition to Recovery.
 - As described in § Section 4.2.5.4, an Electrical Idle condition may be inferred on all Lanes under any one of the following conditions: (i) absence of a Flow Control Update DLLP in any 128 μs window, (ii) absence of a SKP Ordered Set in any of the configured Lanes in any 128 μs window, or (iii) absence of a Flow Control Update DLLP, an Optimized_Update_FC, or a SKP Ordered Set in any of the configured Lanes in any 128 μs window.
 - The clause “if directed” applies to a Port that is instructed by a higher Layer to transition to Recovery including the Retrain Link bit in the Link Control Register being set.
 - The Transmitter may complete any TLP or DLLP in progress.
- Next state is L0s for only the Transmitter if directed to this state and the Transmitter implements L0s. See § Section 4.2.7.6.2.
 - The clause “if directed” applies to a Port that is instructed by a higher Layer to initiate L0s (see § Section 5.4.1.1.1).
 - Note: This is a point where the TX and RX may diverge into different LTSSM states.
- Next state is L0s for only the Receiver if an EIOS is received on any Lane, the Receiver implements L0s, and the Port is not directed to L1 or L2 states by any higher layers. See § Section 4.2.7.6.1.
 - Note: This is a point where the TX and RX may diverge into different LTSSM states.
- Next state is Recovery if an EIOS is received on any Lane, the Receiver does not implement L0s, the Port is not directed to L1 or L2 states by any higher layers, and the EIOS is not expected as part of an L0p transition to a lower width. See § Section 4.2.7.6.1 and § Section 4.2.6.7.
- Next state is L1 :
 - i. If directed
and
 - ii. an EIOS is received on any Lane
and
 - iii. an EIOSQ is transmitted on all Lanes.
 - The clause “if directed” is defined as both ends of the Link having agreed to enter L1 immediately after the condition of both the receipt and transmission of the EIOS (s) is met. A transition to L1 can be initiated by PCI-PM (see § Section 5.3.2.1) or by ASPM (see § Section 5.4.1.3.1).
 - Note: When directed by a higher Layer one side of the Link always initiates and exits to L1 by transmitting the EIOS (s) on all Lanes, followed by a transition to Electrical Idle.⁹⁶ The same Port then

waits for the receipt of an EIOS on any Lane, and then immediately transitions to L1 . Conversely, the side of the Link that first receives the EIOS (s) on any Lane must send an EIOSQ on all Lanes and immediately transition to L1 .

- Next state is L2 :
 - i. If directed and
 - ii. an EIOS is received on any Lane and
 - iii. an EIOSQ is transmitted on all Lanes.
 - The clause “if directed” is defined as both ends of the Link having agreed to enter L2 immediately after the condition of both the receipt and transmission of the EIOS (s) is met (see § Section 5.3.2.3 for more details).
 - Note: When directed by a higher Layer, one side of the Link always initiates and exits to L2 by transmitting EIOS on all Lanes followed by a transition to Electrical Idle.⁹⁷ The same Port then waits for the receipt of EIOS on any Lane, and then immediately transitions to L2 . Conversely, the side of the Link that first receives an EIOS on any Lane must send an EIOSQ on all Lanes and immediately transition to L2 .

4.2.7.6 L0s §

The L0s substate machine is shown in § Figure 4-72 .

4.2.7.6.1 Receiver L0s §

A Receiver must implement L0s if its Port advertises support for L0s , as indicated by the ASPM Support field in the Link Capabilities Register . It is permitted for a Receiver to implement L0s even if its Port does not advertise support for L0s .

4.2.7.6.1.1 Rx_L0s.Entry §

- Next state is Rx_L0s.Idle after a T_{TX-IDLE-MIN} (§ Table 8-7) timeout.
 - Note: This is the minimum time the Transmitter must be in an Electrical Idle condition.

4.2.7.6.1.2 Rx_L0s.Idle §

- Next state is Rx_L0s.FTS if the Receiver detects an exit from Electrical Idle on any Lane of the configured Link.
- Next state is Rx_L0s.FTS after a 100 ms timeout if the current data rate is 8.0 GT/s or higher and the Port’s Receivers do not meet the Z_{RX-DC} specification for 2.5 GT/s (see § Table 8-12). All Ports are permitted to implement the timeout and transition to Rx_L0s.FTS when the data rate is 8.0 GT/s or higher.

96. The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle (V_{TX-CM-DC-ACTIVE-IDLE-DELTA}) specification (see § Table 8-7).

97. The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle (V_{TX-CM-DC-ACTIVE-IDLE-DELTA}) specification (see § Table 8-7).

4.2.7.6.1.3 Rx_L0s.FTS §

- The next state is L0 if a SKP Ordered Set is received in 8b/10b encoding or the SDS Ordered Set is received for 128b/130b encoding on all configured Lanes of the Link.
 - The Receiver must be able to accept valid data immediately after the SKP Ordered Set for 8b/10b encoding.
 - The Receiver must be able to accept valid data immediately after the SDS Ordered Set for 128b/130b encoding.
 - Lane-to-Lane de-skew must be completed before leaving Rx_L0s.FTS .
- Otherwise, next state is Recovery after the N_FTS timeout.
 - When using 8b/10b encoding: The N_FTS timeout shall be no shorter than $40*[N_FTS+3]*UI$ (The 3 * 40 UI is derived from six Symbols to cover a maximum SKP Ordered Set + four Symbols for a possible extra FTS+2 Symbols of design margin), and no longer than twice this amount. When the Extended Synch bit is Set the Receiver N_FTS timeout must be adjusted to no shorter than $40*[2048]*UI$ (2048 FTSS) and no longer than $40*[4096]*UI$ (4096 FTSS). Implementations must take into account the worst case Lane to Lane skew, their design margins, as well as the four to eight consecutive EIE Symbols in speeds other than 2.5 GT/s when choosing the appropriate timeout value within the specification's defined range.
 - When using 128b/130b encoding: The N_FTS timeout shall be no shorter than $130*[N_FTS+5+12+Floor(N_FTS/32)]*UI$ and no longer than twice this amount for 8.0 GT/s and 16.0 GT/s data rates. For 32.0 GT/s and above data rates, the N_FTS timeout shall be no shorter than $130*[N_FTS+10+12+2*Floor(N_FTS/32)]*UI$ and no longer than twice this amount. The 5+Floor(N_FTS/32) accounts for the first EIEOS , the last EIEOS , the SDS, the periodic EIEOS and an additional EIEOS in case an implementation chooses to send two EIEOS followed by an SDS when N_FTS is divisible by 32 for 8.0 GT/s and 16.0 GT/s data rates and correspondingly doubled for the 32.0 GT/s and higher data rates. The 12 is there to account for the number of SKP Ordered Sets that will be transmitted if the Extended Synch bit is Set. When the Extended Synch bit is Set, the timeout should be the same as the normal case with N_FTS equal to 4096.
 - The Transmitter must also transition to Recovery , but is permitted to complete any TLP or DLLP in progress.
 - It is recommended that the N_FTS field be increased when transitioning to Recovery to prevent future transitions to Recovery from Rx_L0s.FTS .

4.2.7.6.2 Transmitter L0s §

A Transmitter must implement L0s if its Port advertises support for L0s , as indicated by the ASPM Support field in the Link Capabilities Register . It is permitted for a Transmitter to implement L0s even if its Port does not advertise support for L0s .

4.2.7.6.2.1 Tx_L0s.Entry §

- Transmitter sends an EIOSQ and enters Electrical Idle.
 - The DC common mode voltage must be within specification by $T_{TX-IDLE-SET-TO-IDLE}$.⁹⁸

98. The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).

- Next state is Tx_L0s.Idle after a $T_{TX-IDLE-MIN}$ (§ Table 8-7) timeout.

4.2.7.6.2.2 Tx_L0s.Idle §

- Next state is Tx_L0s.FTS if directed.

IMPLEMENTATION NOTE: INCREASE OF N_FTS DUE TO TIMEOUT IN RX_L0s.FTS §

The Transmitter sends the N_FTS Fast Training Sequences by going through Tx_L0s.FTS substate to enable the Receiver to reacquire its bit and Symbol lock or Block alignment. In the absence of the N_FTS Fast Training Sequence , the Receiver will timeout in Rx_L0s.FTS substate and may increase the N_FTS number it advertises in the Recovery state.

4.2.7.6.2.3 Tx_L0s.FTS §

- Transmitter must send N_FTS Fast Training Sequences on all configured Lanes.
 - Four to eight EIE Symbols must be sent prior to transmitting the N_FTS (or 4096 if the Extended Synch bit is Set) number of FTS in 5.0 GT/s data rates. An EIEOSQ must be sent prior to transmitting the N_FTS (or 4096 if the Extended Synch bit is Set) number of FTS with 128b/130b encoding. In 2.5 GT/s speed, up to one full FTS may be sent before the N_FTS (or 4096 if the Extended Synch bit is Set) number of FTSS are sent.
 - SKP Ordered Sets must not be inserted before all FTSS as defined by the agreed upon N_FTS parameter are transmitted.
 - If the Extended Synch bit is Set, the Transmitter must send 4096 Fast Training Sequences , inserting SKP Ordered Sets according to the requirements in § Section 4.2.5.6 .
- When using 8b/10b encoding, the Transmitter must send a single SKP Ordered Set on all configured Lanes.
- When using 128b/130b encoding, the Transmitter must send one EIEOSQ followed by one SDS Ordered Set on all configured Lanes. Note: The first Symbol transmitted on Lane 0 after the SDS Ordered Set is the first Symbol of the Data Stream.
- Next state must be L0 , after completing the above required transmissions.

IMPLEMENTATION NOTE: NO SKP ORDERED SET REQUIREMENT WHEN EXITING LOS AT 16.0 GT/S OR HIGHER DATA RATES §

Unlike in other LTSSM states, when exiting Tx_L0s.FTS no Control SKP Ordered Set is transmitted before transmitting the SDS. This results in the Data Parity information associated with the last portion of the previous datastream being discarded. Not sending the Control SKP Ordered Set reduces complexity and improves exit latency.

Base 6.4 vs Base 6.3

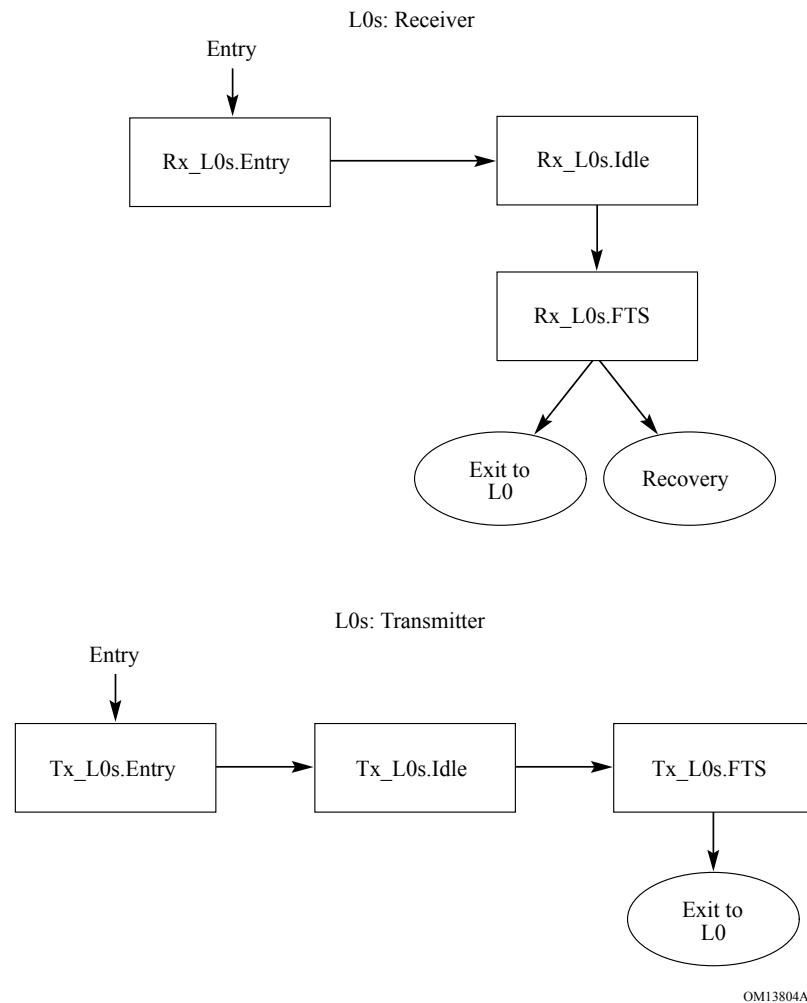


Figure 4-72 L0s Substate Machine §

4.2.7.7 L1 §

The L1 substate machine is shown in § Figure 4-73.

4.2.7.7.1 L1.Entry §

- All configured Transmitters are in Electrical Idle.
 - The DC common mode voltage must be within specification by $T_{TX-IDLE-SET-TO-IDLE}$.
- The next state is L1.Idle after a $T_{TX-IDLE-MIN}$ (§ Table 8-7) timeout.
 - Note: This guarantees that the Transmitter has established the Electrical Idle condition.

4.2.7.7.2 L1.Idle §

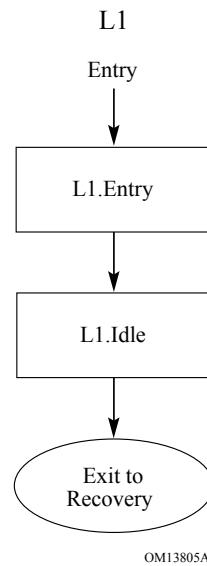
- Transmitter remains in Electrical Idle.

- The DC common mode voltage must be within specification, except as allowed by L1 PM Substates , when applicable.⁹⁹
- A substate of L1 is entered when the conditions for L1 PM Substates are satisfied (see § Section 5.5).
 - The L1 PM Substate must be L1.0 when L1.Idle is entered or exited.
- Next state is Recovery if exit from Electrical Idle is detected on any Lane of a configured Link, or directed after remaining in this substate for a minimum of 40 ns in speeds other than 2.5 GT/s.
 - Ports are not required to arm the Electrical Idle exit detectors on all Lanes of the Link.
 - Note: A minimum stay of 40 ns is required in this substate in speeds other than 2.5 GT/s to account for the delay in the logic levels to arm the Electrical Idle detection circuitry in case the Link enters L1 and immediately exits the L1 state.
 - A Port is allowed to set the directed_speed_change variable to 1b following identical rules described in L0 for setting this variable. When making such a transition, the changed_speed_recovery variable must be reset to 0b. A Port may also go through Recovery back to L0 and then set the directed_speed_change variable to 1b on the transition from L0 to Recovery .
 - A Port is also allowed to enter Recovery from L1 if directed to change the Link width. The Port must follow identical rules for changing the Link width as described in the L0 state.
- Next state is Recovery after a 100 ms timeout if the current data rate is 8.0 GT/s or higher and the Port's Receivers do not meet the Z_{RX-DC} specification for 2.5 GT/s. All Ports are permitted, but not encouraged, to implement the timeout and transition to Recovery when the data rate is 8.0 GT/s or higher.
 - This timeout is not affected by the L1 PM Substates mechanism.

IMPLEMENTATION NOTE: 100 MS TIMEOUT IN L1 §

Ports that meet the Z_{RX-DC} specification for 2.5 GT/s while in the L1.Idle state and are therefore not required to implement the 100 ms timeout and transition to Recovery should avoid implementing it, since it will reduce the power savings expected from the L1 state.

99. The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).



OM13805A

Figure 4-73 L1 Substate Machine §

4.2.7.8 L2 §

The L2 substate machine is shown in § Figure 4-74.

4.2.7.8.1 L2.Idle §

- All Receivers must meet the Z_{RX-DC} specification for 2.5 GT/s within 1 ms (see § Table 8-12).
- All configured Transmitters must remain in Electrical Idle for a minimum time of $T_{TX-IDLE-MIN}$.
 - The DC common mode voltage does not have to be within specification.
 - The Receiver needs to wait a minimum of $T_{TX-IDLE-MIN}$ to start looking for Electrical Idle Exit.
- For Downstream Lanes:
 - For all Downstream Ports, the next state is Detect if a Beacon is received on at least Lane 0 or if directed.
 - Main power must be restored before entering Detect .
 - The clause “if directed” is defined as a higher layer decides to exit to Detect .
 - For a Switch, if a Beacon is received on at least Lane 0 of any of its Downstream Ports and the Upstream Port is in L2.Idle , the Upstream Port must be directed to L2.TransmitWake .
- For Upstream Lanes:
 - The next state is Detect if Electrical Idle Exit is detected on any predetermined set of Lanes.
 - The predetermined set of Lanes must include but is not limited to any Lane which has the potential of negotiating to Lane 0 of a Link. For multi-Lane Links the number of Lanes in the predetermined set must be greater than or equal to two.
 - A Switch must transition any Downstream Lanes to Detect .
 - Next state is L2.TransmitWake for an Upstream Port if directed to transmit a Beacon.

- Note: Beacons may only be transmitted on Upstream Ports in the direction of the Root Complex.

4.2.7.8.2 L2.TransmitWake §

This state only applies to Upstream Ports.

- Transmit the Beacon on at least Lane 0.
- Next state is Detect if Electrical Idle exit is detected on any Upstream Port's Receiver that is in the direction of the Root Complex.
 - Note: Power is guaranteed to be restored when Upstream Receivers see Electrical Idle exited, but it may also be restored prior to Electrical Idle being exited.

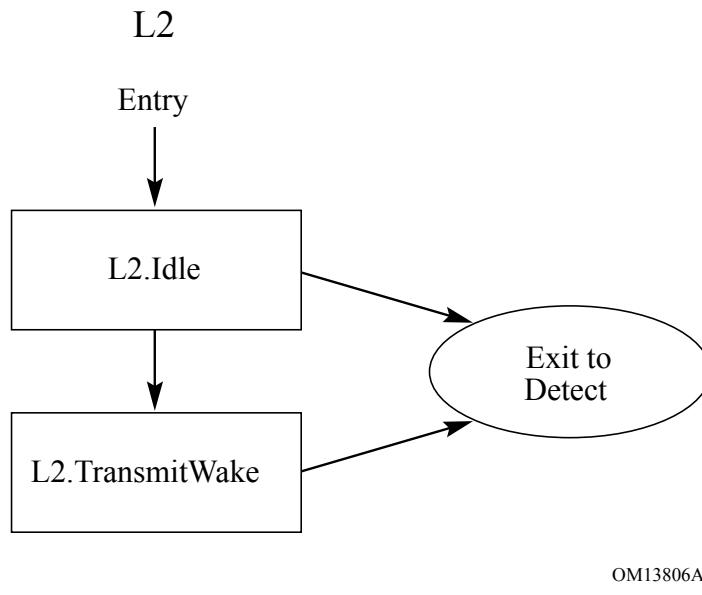


Figure 4-74 L2 Substate Machine §

4.2.7.9 Disabled §

- It is recommended to Clear LinkUp upon entry to Disabled, without waiting for the EIOSQ to be transmitted or the EIOS to be received.
- All Lanes transmit 16 to 32 TS1 Ordered Sets with Disable_Link_Request asserted and then transition to Electrical Idle.
 - An EIOSQ must be sent prior to entering Electrical Idle.
 - The DC common mode voltage does not have to be within specification.¹⁰⁰
- If an EIOSQ was transmitted and an EIOS was received on any Lane (even while transmitting TS1 with Disable_Link_Request asserted), then:
 - LinkUp = 0b (False), unless already Cleared, as recommended above.
 - At this point, the Lanes are considered Disabled.

¹⁰⁰. The common mode being driven does need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle (V_{TX-CM-DC-ACTIVE-IDLE-DELTA}) specification (see § Table 8-7).

- For Upstream Ports: All Receivers must meet the Z_{RX-DC} specification for 2.5 GT/s within 1 ms (see § Table 8-12).
- For Upstream Ports: The next state is Detect when Electrical Idle exit is detected on at least one Lane.
- For Downstream Ports: The next state is Detect when directed (e.g., when the Link Disable bit is reset to 0b by software).
- For Upstream Ports: If no EIOS is received after a 2 ms timeout, the next state is Detect.

4.2.7.10 Loopback §

The Loopback substate machine is shown in § Figure 4-75 .

4.2.7.10.1 Loopback.Entry §

- LinkUp = 0b (False)
- The Link and Lane numbers received in the TS1 or TS2 Ordered Sets are ignored by the Receiver while in this substate.
- Loopback Lead requirements:
 - If the Loopback Lead was directed, in an implementation specific manner, to perform a 64.0 GT/s equalization procedure on one active Lane, to be referred to as the ‘Lane under test’, before entering Loopback.Entry with LinkUp set to 0b, the perform_equalization_for_loopback_64GT variable must be set to 1b.
 - If Loopback.Entry was entered from Recovery.Equalization and the current data rate is 32.0 GT/s and perform_equalization_for_loopback_64GT is 1b and the equalization_done_64GT_data_rate variable is 0b, determine the highest common data rate of the data rates supported by the Lead and the highest data rates advertised by the Loopback Follower in the TS1s that it transmitted in Phase 1 of Recovery.Equalization on the ‘Lane Under Test’. If the highest common data rate is 64.0 GT/s, transmit 16 consecutive TS1 Ordered Sets with Loopback_Request asserted, followed by an EIOSQ, and then transition to Electrical Idle for 1 ms. During the period of Electrical Idle, change the data rate to 64.0 GT/s.

The 16 consecutive TS1 Ordered Sets transmitted on the Lane under test prior to the rate change to 64.0 GT/s must have the bits listed below as follows:

- The Enhanced Link Behavior Control bits must be set to 01b. The equalization procedure must be performed on the same ‘Lane under test’ for both equalization procedures.
- The Transmit Modified Compliance Pattern in Loopback bit must be set to 1b if the Loopback Follower is required to transmit the Modified Compliance Pattern on the Lanes that are not under test.
- If Loopback.Entry was entered from Configuration.Linkwidth.Start, the highest common data rate is 64.0 GT/s when LinkUp = 0b, the Loopback Lead intends to operate at 64.0 GT/s and the Loopback Lead has prior knowledge that the Loopback Follower also supports 64.0 GT/s. Otherwise, determine the highest common data rate of the data rates supported by the Lead and the data rates received in two consecutive TS1 or TS2 Ordered Sets on any active Lane at the time the transition to Loopback.Entry occurred. If the current data rate is not the highest common data rate:
 - Transmit 16 consecutive TS1 Ordered Sets with Loopback_Request asserted, followed by an EIOSQ, and then transition to Electrical Idle for 1 ms. During the period of Electrical Idle, if the perform_equalization_for_loopback_64GT variable is 1b and the highest common data

rate is 64.0 GT/s, change the data rate to 32.0 GT/s. Otherwise, change the data rate to the highest common data rate.

- If LinkUp is 0b, the Supported Link Speeds field of the Data Rate Identifier must use the Flit Mode data rate encoding if the Flit Mode Supported field in the PCI Express Capabilities Register is 1b and 64.0 GT/s is supported and the consecutive TS2 Ordered Sets received on the last transition from Polling.Configuration had the Flit Mode Supported bit set to 1b.
- The Loopback Lead may be directed, in an implementation specific manner, to perform a 32.0 GT/s equalization procedure on one active Lane, to be referred to as the ‘Lane under test’, before entering Loopback.Entry. If the highest common data rate is 32.0 GT/s, the equalization_done_32GT_data_rate variable is 0b, and the equalization procedure is to be executed, the 16 consecutive TS1 Ordered Sets transmitted on the Lane under test prior to the data rate change to the highest common data rate must have the bits listed below as follows:
 - The Enhanced Link Behavior Control bits must be set to 01b.
 - The Transmit Modified Compliance Pattern in Loopback bit must be set to 1b if the Loopback Follower is required to transmit the Modified Compliance Pattern on the Lanes that are not under test.

IMPLEMENTATION NOTE: LANE UNDER TEST USAGE EXPECTATIONS §

The method whereby one active Lane is defined as the ‘Lane under test’ and is affected by the NEXT/FEXT aggressor Lanes (see § Section 8.5.1.1) so that measurements are performed on the Lane under test (after a speed change and completion of an equalization procedure at a rate of 32.0 GT/s or above) is defined for system configurations where a test apparatus, such as (but not limited to) a BERT, acts as the Loopback Lead. In such a system configuration, the expectation of this test method is that the Loopback Lead is able to provide the necessary stimulus for the state traversals and protocol negotiations required to establish and exercise the ‘Lane under test’ without any specific guidance from this specification.

This test mode is only defined for use while LinkUp = 0b (prior to entering Loopback.Entry). The test apparatus must be cognizant of the capabilities of the ‘Lane under test’. A mechanism to “re-do” equalization when this test procedure is performed is not defined. The Loopback Lead must only send TS1s with Flit Mode Encoding in Loopback if it received TS2s with the Flit Mode Supported bit set to 1b in Polling.Configuration - these TS2s indicate that the Loopback Follower will properly interpret TS1s with Flit Mode Data Rate Encoding.

- If the highest common data rate is 5.0 GT/s, the follower’s transmitter de-emphasis is controlled by setting the Selectable De-emphasis bit of the transmitted TS1 Ordered Sets to the desired value (1b = -3.5 dB, 0b = -6 dB).
- For data rates of 5.0 GT/s and above, the Lead is permitted to choose its own transmitter settings in an implementation specific manner, regardless of the settings it transmitted to the follower.
- Note: If Loopback is entered after LinkUp has been set to 1b, it is possible for one Port to enter Loopback from Recovery and the other to enter Loopback from Configuration. The Port that entered from Configuration might attempt to change data rate while the other

Port does not. If this occurs, the results are undefined. The test set-up must avoid such conflicting directed clauses.

- Transmit TS1 Ordered Sets with Loopback_Request asserted.
 - If Loopback.Entry was entered from Recovery.Equalization, the EC field of the transmitted TS1 Ordered Sets must be set to 00b.
 - The Lead is also permitted to assert Compliance_Receive_Request of TS1 Ordered Sets transmitted in Loopback.Entry, including those transmitted before a data rate change. If it asserts Compliance_Receive_Request, it must not deassert it again while in the Loopback.Entry state. This usage model might be helpful for test and validation purposes when one or both Ports have difficulty obtaining bit lock, Symbol lock, or Block alignment after a data rate change. The ability to assert Compliance_Receive_Request is implementation specific.
- Next state is Recovery.Equalization if the data rate was changed to 32.0 or 64.0 GT/s and 16 consecutive TS1 Ordered Sets were sent on any Lane with the Enhanced Link Behavior Control bits set to 01b.
 - The perform_equalization_for_loopback variable is set to 1b.
- Next state is Loopback.Active after 2 ms if Compliance_Receive_Request of the transmitted TS1 Ordered Sets is asserted.
- Next state is Loopback.Active if Loopback.Entry was entered from Recovery.Equalization and the Lane under test receives two consecutive TS1 Ordered Sets with Loopback_Request asserted.
- Next state is Loopback.Active if Compliance_Receive_Request of the transmitted TS1 Ordered Sets is deasserted and an implementation specific set of Lanes receive two consecutive TS1 Ordered Sets with Loopback_Request asserted.

If the data rate was changed and the equalization procedure was not performed at either 32.0 GT/s or 64.0 GT/s, the Lead must take into account the amount of time the follower can be in Electrical Idle and transmit a sufficient number of TS1 Ordered Sets for the follower to acquire Symbol lock or Block alignment before proceeding to Loopback.Active. These TS1 Ordered Sets may be used to specify Transmitter settings for the Loopback Follower when the data rate of operation is 8.0 GT/s or above by setting the value of the EC field appropriately for the follower's Port direction (10b or 11b) and requesting a preset or a set of valid coefficients.

IMPLEMENTATION NOTE: LANE NUMBERING WITH 128B/130B ENCODING IN LOOPBACK §

If the current data rate uses 128b/130b encoding and Lane numbers have not been negotiated, it is possible that the Lead and follower will not be able to decode received information because their Lanes are using different scrambling LFSR seed values (since the LFSR seed values are determined by the Lane numbers). This situation can be avoided by allowing the Lead and follower to negotiate Lane numbers before directing the Lead to Loopback, directing the Lead to assert Compliance_Receive_Request during Loopback.Entry, or by using some other method of ensuring that the LFSR seed values match.

- Next state is Loopback.Exit after an implementation specific timeout of less than 100 ms.
- Loopback Follower requirements:

- If Loopback.Entry was entered from Configuration.Linkwidth.Start with LinkUp set to 0b, the TS1 Ordered Sets that directed the follower to this state advertised 64.0 GT/s support and had Enhanced Link Behavior Control set to 01b and the follower intends to operate at 64.0 GT/s, perform_equalization_for_loopback_64GT must be set to 1b.
- If Loopback.Entry was entered from Recovery.Equalization and the current data rate is 32.0 GT/s and perform_equalization_for_loopback_64GT is 1b and the equalization_done_64GT_data_rate variable is 0b, and 64.0 GT/s is advertised by the Loopback Follower in the TS1s that it transmitted in Phase 1 of Recovery.Equalization on the ‘Lane under test’:
 - Transmit an EIOSQ, and then transition to Electrical Idle for 2 ms. During the period of Electrical Idle, change the data rate to 64.0 GT/s.
 - If the Loopback Follower is a Downstream Port, transmit 16 consecutive TS1 Ordered Sets on the ‘Lane under test’ prior to transmitting the EIOSQ. The TS1 Ordered Sets must have the Equalization Control bits set to 00b.
 - Next state is Recovery.Equalization.
 - The ‘Lane under test’, perform_equalization_for_loopback variable, transmit_modified_compliance_pattern_in_loopback variable and the Link and Lane numbers for both the 32.0 GT/s and 64.0 GT/s equalization procedures must be the same.
- If Loopback.Entry was entered from Configuration.Linkwidth.Start, determine the highest common data rate of the data rates supported by the follower and the data rates received in the two consecutive TS1 Ordered Sets that directed the follower to this state. If the current data rate is not the highest common data rate:
 - Transmit an EIOSQ, and then transition to Electrical Idle for 2 ms. During the period of Electrical Idle, if the perform_equalization_for_loopback_64GT variable is 1b and the highest common data rate is 64.0 GT/s, change the data rate to 32.0 GT/s. Otherwise, change the data rate to the highest common data rate.
 - If operating in full swing mode and the highest common data rate is 5.0 GT/s, set the transmitter’s de-emphasis to the setting specified by the Selectable De-emphasis bit received in the TS1 Ordered Sets that directed the follower to this state. The de-emphasis is -3.5 dB if the Selectable De-emphasis bit was 1b, and it is -6 dB if the Selectable De-emphasis bit was 0b.
 - If the highest common data rate is 8.0 GT/s or higher and EQ TS1 Ordered Sets directed the follower to this state, set the transmitter to the settings specified by the Preset field of the EQ TS1 Ordered Sets. See § Section 4.2.4.2. If the highest common data rate is 8.0 GT/s or higher but standard TS1 Ordered Sets directed the follower to this state, the follower is permitted to use its default transmitter settings.
- Next state is Recovery.Equalization if Loopback.Entry was entered from Configuration.Linkwidth.Start, the highest common data rate is 32.0 GT/s or higher and the Enhanced Link Behavior Control bits of the TS1 Ordered Sets that directed the follower to this state were 01b.
 - The **perform_equalization_for_loopback** variable is set to 1b.
 - The **transmit_modified_compliance_pattern_in_loopback** variable is set to 1b if the Transmit Modified Compliance Pattern in Loopback bit is Set to 1b in the TS1 Ordered Sets that directed the follower to this state.
 - When Recovery.Equalization is entered from Loopback.Entry, the Lane that received two consecutive TS1 Ordered Sets with the Enhanced Link Behavior Control bits set to 01b in Configuration.Linkwidth.Start is the Lane under test for the purposes of Loopback and Recovery.Equalization.

- The Loopback Follower must select a valid Link number in an implementation specific manner. (See § Section 4.2.2.4 for more on assigning Lane numbers and other Scrambler requirements.) The test measurement equipment that facilitates this state transition must ensure, in an implementation specific manner, that it uses a matching Lane number and LFSR seed value.
- Next state is Loopback.Active if Compliance_Receive_Request in the TS1 Ordered Sets that directed the follower to this state was asserted.
 - The follower's transmitter does not need to transition to transmitting looped-back data on any boundary, and it is permitted to truncate any Ordered Set in progress.
- Else, the follower transmits TS1 Ordered Sets with Link and Lane numbers set to PAD.
 - If Loopback.Entry was entered from Recovery.Equalization :
 - The EC field of the transmitted TS1 Ordered Sets must be set to 00b.
 - The next state is Loopback.Active after two consecutive TS1 Ordered Sets with Loopback_Request asserted are received by the Lane under test.
 - Else, the next state is Loopback.Active if one of the following conditions is true:
 - The data rate is 2.5 GT/s or 5.0 GT/s and Symbol lock is obtained on all active Lanes.
 - The data rate is 8.0 GT/s or higher and two consecutive TS1 Ordered Sets are received on all active Lanes. The equalization settings specified by the received TS1 Ordered Sets must be evaluated and applied to the transmitter if the value of the EC field is appropriate for the follower's Port direction (10b or 11b) and the requested setting is a preset or a set of valid coefficients. (Note: This is the equivalent behavior for the Recovery.Equalization state.) Optionally, the follower can accept both EC field values. If the settings are applied, they must take effect within 500 ns of being received, and they must not cause the transmitter to violate any electrical specification for more than 1 ns. Unlike Recovery.Equalization, the new settings are not reflected in the TS1 Ordered Sets that the follower transmits.
 - When using 8b/10b encoding, the follower's transmitter must transition to transmitting looped-back data on a Symbol boundary, but it is permitted to truncate any Ordered Set in progress. When using 128b/130b or 1b/1b encoding, the follower's transmitter does not need to transition to transmitting looped-back data on any boundary and is permitted to truncate any Ordered Set in progress.

4.2.7.10.2 Loopback.Active §

- The Loopback Lead must send valid encoded data. The Loopback Lead must not transmit EIOS as data until it wants to exit Loopback. When operating with 128b/130b encoding, Loopback Leads must follow the requirements of § Section 4.2.2.6 .
- When operating with 1b/1b encoding, the Loopback Lead must follow the requirements of § Section 4.2.3.1 , when it starts a Data Stream. The Loopback Lead is permitted to substitute the CRC and FEC bytes with any bytes of its choice. If the Link width has been negotiated by entering L0 since leaving the Detect State, that width is to be used; else the width must be assumed to be x1.
- A Loopback Follower that entered Loopback from Recovery.Equalization must transmit the Modified Compliance Pattern on all Lanes that detected Receivers in Detect.Active but are not under test if the transmit_modified_compliance_pattern_in_loopback variable is set to 1b, otherwise those Lanes must be transitioned into Electrical Idle. The Lane under test must follow Loopback Follower rules described below. State transitions must be based only on Link activity on the Lane under test.

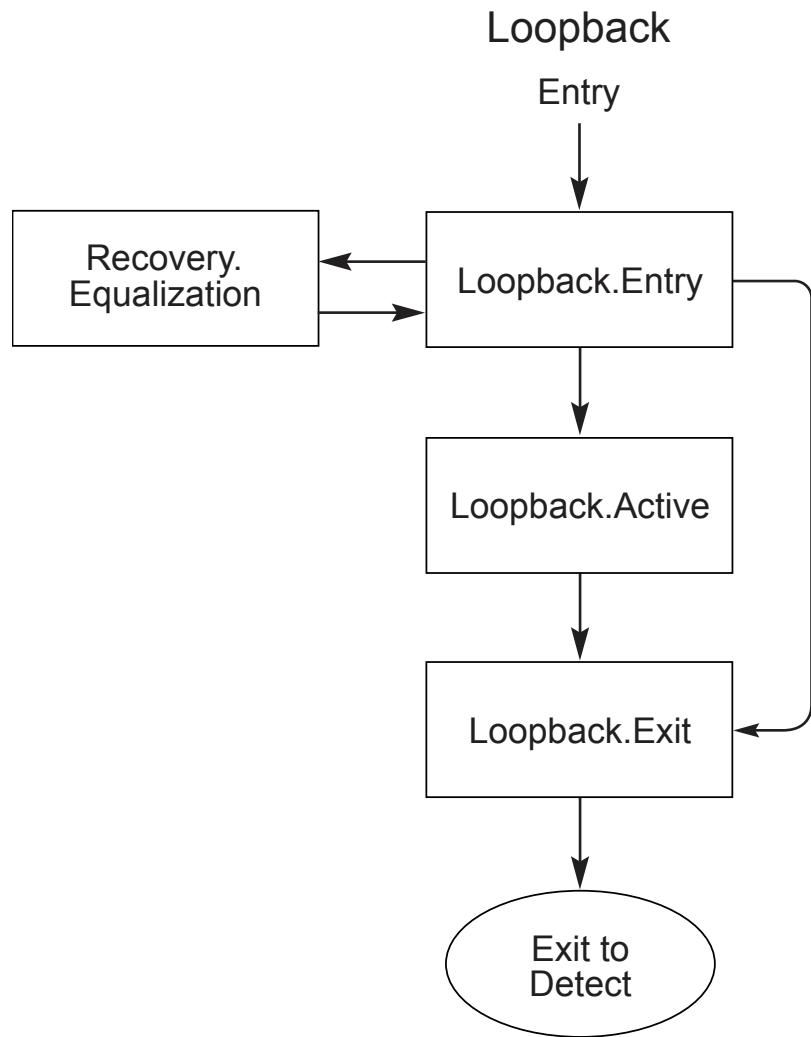
- A Loopback Follower is required to retransmit the received encoded information as received, with the polarity inversion determined in Polling applied, while continuing to perform clock tolerance compensation:
 - When operating with 1b/1b encoding: The Loopback Follower must track when the Data Stream transition happens and subsequently distinguish an Ordered Set from Data Stream bytes as defined in § Section 4.2.3.1 , using the Ordered Set insertion interval of § Table 4-32 as follows:
 - If the Link width has been negotiated by entering L0 since leaving the Detect State, that width is to be used; else the width must be assumed to be x1
 - For Upstream Lanes, the SRIS mode is determined from the TS1 Ordered Set (Symbol 4, Bit 7) received in the Configuration state whereas for Downstream Lanes it comes from the SRIS Clocking bit in the Link Control Register .
 - SKPs must be added or deleted on a per-Lane basis as outlined in § Section 4.2.8 with the exception that SKPs do not have to be simultaneously added or removed across Lanes of a configured Link.
 - For 8b/10b encoding, if a SKP Ordered Set retransmission requires adding a SKP Symbol to accommodate timing tolerance correction, the SKP Symbol is inserted in the retransmitted Symbol stream anywhere adjacent to a SKP Symbol in the SKP Ordered Set following the COM Symbol. The inserted SKP Symbol must be of the same disparity as the received SKPs Symbol(s) in the SKP Ordered Set .
 - For 8b/10b encoding, if a SKP Ordered Set retransmission requires dropping a SKP Symbol to accommodate timing tolerance correction, the SKP Symbol is simply not retransmitted.
 - For 128b/130b encoding, if a SKP Ordered Set retransmission requires adding SKP Symbols to accommodate timing tolerance correction, four SKP Symbols are inserted in the retransmitted Symbol stream prior to the SKP_END Symbol in the SKP Ordered Set .
 - For 128b/130b encoding, if a SKP Ordered Set retransmission requires dropping SKP Symbols to accommodate timing tolerance correction, four SKP Symbols prior to the SKP_END Symbol in the SKP Ordered Set are simply not retransmitted.
 - For 1b/1b encoding, if a SKP Ordered Set retransmission requires adding SKP Symbols to accommodate timing tolerance correction, eight SKP Symbols are inserted in the retransmitted Symbol stream prior to the SKP_END Symbol in the SKP Ordered Set .
 - For 1b/1b encoding, if a SKP Ordered Set retransmission requires dropping SKP Symbols to accommodate timing tolerance correction, eight SKP Symbols prior to the SKP_END Symbol in the SKP Ordered Set are simply not retransmitted.
 - No modifications of the received encoded data (except for polarity inversion determined in Polling) are allowed by the Loopback Follower even if it is determined to be an invalid encoding (i.e., no legal translation to a control or data value possible for 8b/10b encoding or invalid Sync Header or invalid Ordered Set for 128b/130b encoding).
- Next state of the Loopback Follower is Loopback.Exit if one of the following conditions apply:
 - If directed or if four consecutive EIOSs are received on any Lane. The Requirements for detecting an EIOS are specified in § Section 4.2.5.3 .
 - Optionally, if current Link speed is 2.5 GT/s and an EIOS is received or Electrical Idle is detected/inferred on any Lane.
 - Note: As described in § Section 4.2.5.4 , an Electrical Idle condition may be inferred if any of the configured Lanes remained electrically idle continuously for 128 µs by not detecting an exit from Electrical Idle in the entire 128 µs window.
 - A Loopback Follower must be able to detect an Electrical Idle condition on any Lane within 1 ms of the EIOS being received by the Loopback Follower .

- Note: During the time after an EIOS is received and before Electrical Idle is actually detected by the Loopback Follower, the Loopback Follower may receive a bit stream that is undefined by the encoding scheme, which may be looped back by the transmitter.
- The $T_{TX-IDLE-SET-TO-IDLE}$ parameter does not apply in this case since the Loopback Follower may not even detect Electrical Idle until as much as 1 ms after the EIOS.
- The next state of the Loopback Lead is Loopback.Exit if directed.

4.2.7.10.3 Loopback.Exit §

- The Loopback Lead sends an EIOS for Ports that support only the 2.5 GT/s data rate and eight consecutive EIOSs for Ports that support greater than 2.5 GT/s data rate, and optionally for Ports that only support the 2.5 GT/s data rate, irrespective of the current Link speed, and enters Electrical Idle on all Lanes for 2 ms.
 - The Loopback Lead must transition to a valid Electrical Idle condition¹⁰¹ on all Lanes within $T_{TX-IDLE-SET-TO-IDLE}$ after sending the last EIOS.
 - Note: The EIOS can be useful in signifying the end of transmit and compare operations that occurred by the Loopback Lead. Any data received by the Loopback Lead after any EIOS is received should be ignored since it is undefined.
- The Loopback Follower must enter Electrical Idle on all Lanes for 2 ms.
 - Before entering Electrical Idle the Loopback Follower must Loopback all Symbols that were received prior to detecting Electrical Idle. This ensures that the Loopback Lead may see the EIOS to signify the logical end of any Loopback send and compare operations.
- The next state of the Loopback Lead and Loopback Follower is Detect.

¹⁰¹. The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see § Table 8-7).



OM13807C

Figure 4-75 Loopback Substate Machine §

4.2.7.11 Hot Reset §

- Lanes that were directed by a higher Layer to initiate Hot Reset:
 - All Lanes in the configured Link transmit TS1 Ordered Sets with Hot_Reset_Request asserted and the configured Link and Lane numbers.
 - If two consecutive TS1 Ordered Sets are received on any Lane with Hot_Reset_Request asserted and configured Link and Lane numbers, then:
 - LinkUp = 0b (False)
 - If no higher Layer is directing the Physical Layer to remain in Hot Reset, the next state is Detect
 - Otherwise, all Lanes in the configured Link continue to transmit TS1 Ordered Sets with Hot_Reset_Request asserted and the configured Link and Lane numbers.
 - Otherwise, after a 2 ms timeout next state is Detect.

- Lanes that were not directed by a higher Layer to initiate Hot Reset (i.e., received two consecutive TS1 Ordered Sets with Hot_Reset_Request asserted on any configured Lanes):
 - LinkUp = 0b (False)
 - If any Lane of an Upstream Port of a Switch receives two consecutive TS1 Ordered Sets with Hot_Reset_Request asserted, all configured Downstream Ports must transition to Hot Reset as soon as possible.
 - Any optional crosslinks on the Switch are an exception to this rule and the behavior is system specific.
 - All Lanes in the configured Link transmit TS1 Ordered Sets with Hot_Reset_Request asserted and the configured Link and Lane numbers.
 - If two consecutive TS1 Ordered Sets were received with Hot_Reset_Request asserted and the configured Link and Lane numbers, the state continues to be Hot Reset and the 2 ms timer is reset.
 - Otherwise, the next state is Detect after a 2 ms timeout.

Generally, Lanes of a Downstream or optional crosslink Port will be directed to Hot Reset, and Lanes of an Upstream or optional crosslink Port will enter Hot Reset by receiving two consecutive TS1 Ordered Sets with Hot_Reset_Request asserted on any configured Lanes, from Recovery.Idle state.

4.2.8 Clock Tolerance Compensation §

SKP Ordered Sets (defined in § Section 4.2.8.1, § Section 4.2.8.2, and § Section 4.2.8.3) are used to compensate for differences in frequencies between bit rates at two ends of a Link. The Receiver Physical Layer logical sub-block must include elastic buffering which performs this compensation. The interval between SKP Ordered Set transmissions is derived from the Transmit, Receiver, and Refclk specifications specified in § Table 8-6, § Table 8-12, and § Table 8-19.

The specification supports shared reference clocking architectures (common Refclk) where there is no difference between the Tx and Rx Refclk rates, and two kinds of reference clocking architectures where the Tx and Rx Refclk rates differ. Separate Reference Clocks With No SSC - **SRNS**, and Separate Reference Clocks with Independent SSC - **SRIS**. The maximum difference with SRNS is 600 ppm which can result in a clock shift once every 1666 clocks. The maximum difference with SRIS is 5600 ppm which can result in a clock shift every 178 clocks.

Specific form factor specifications are permitted to require the use of only SRIS, only SRNS, or to provide a mechanism for clocking architecture selection. Upstream Ports are permitted to implement support for any combination of SRIS and SRNS (including no support for either), but must conform to the requirements of any associated form factor specification. Downstream Ports supporting SRIS must also support SRNS unless the port is only associated with a specific form factor(s) which modifies these requirements. Port configuration to satisfy the requirements of a specific associated form factor is implementation specific. Specific guidance for form factor specifications is provided in § Section 8.6.8.

If the Receiver is capable of operating with SKP Ordered Sets being generated at the rate used in SRNS even though the Port is running in SRIS, the Port is permitted to Set its bit for the appropriate data rate in the Lower SKP OS Reception Supported Speeds Vector field of the Link Capabilities 2 Register. If the transmitter is capable of operating with SKP Ordered Sets being generated at the rate used in SRNS even though the Port is running in SRIS, the Port is permitted to Set its bit for the appropriate data rate in the Lower SKP OS Generation Supported Speeds Vector field of the Link Capabilities 2 register. System software must check that the bit is Set in the Lower SKP OS Reception Supported Speeds Vector field before setting the appropriate data rate's bit in the link partner's Enable Lower SKP OS Generation Vector field of the Link Control 3 Register. Any software transparent Extension Devices (such as a repeater) present on a Link must also support lower SKP OS Generation for system software to set the bit in the Enable Lower SKP OS Generation Vector field. Software determination of such support in such extension devices is implementation specific. When the bit for the data rate that the link is running in is Set in the Enable Lower SKP OS Generation Vector field, the transmitter schedules SKP Ordered Set

generation in L0 at the rate used in SRNS, regardless of which clocking architecture the link is running in. In other LTSSM states, SKP Ordered Set scheduling is at the appropriate rate for the clocking architecture.

Components supporting SRIS may need more entries in their elastic buffers than designs supporting SRNS only. This requirement takes into account the extra time it may take to schedule a SKP Ordered Set if the latter falls immediately after a maximum payload sized packet.

4.2.8.1 SKP Ordered Set for 8b/10b Encoding §

When using 8b/10b encoding, a transmitted SKP Ordered Set is a COM Symbol followed by three SKP Symbols, except as is allowed for a Loopback Follower in the Loopback.Active LTSSM state. A received SKP Ordered Set is a COM Symbol followed by one to five SKP Symbols. See § Section 4.3.7.7 for Retimer rules on SKP Ordered Set modification.

4.2.8.2 SKP Ordered Set for 128b/130b Encoding §

When using 128b/130b encoding, a transmitted SKP Ordered Set is 16 Symbols, and a received SKP Ordered Set can be 8, 12, 16, 20, or 24 Symbols. See § Section 4.3.7.7 for Retimer rules on SKP Ordered Set modification.

There are two SKP Ordered Set formats defined for 128b/130b encoding as shown in § Table 4-64 and § Table 4-65. Both formats contain one to five groups of four SKP Symbols followed by a final group of four Symbols indicated by a SKP-END or SKP-END-CTL Symbol. When operating at 8.0 GT/s in Non-Flit Mode, only the Standard SKP Ordered Set is used. When operating at 16.0 GT/s or 32.0 GT/s in Non-Flit Mode, both the Standard and Control SKP Ordered Sets are used. When operating at 8.0, 16.0, 32.0 GT/s in Flit Mode, both the Standard SKP Ordered Sets and Control SKP Ordered Sets are used. All statements in this specification that do not refer to a specific SKP Ordered Set format apply to both formats. ↑↓When a SKP Ordered Set is transmitted, all Lanes must transmit the same format↑
↑↓Additional requirements for when each type↑
↑↓SKP Ordered Set is sent are defined in↑
↑↓SKP Ordered Set – all Lanes must transmit the Standard SKP Ordered Set, or all Lanes must
transmit the Control SKP Ordered Set↓↑↓\$ Section 4.2.8.4↑.

Errata: Base 6.3
B845△◀▷

The **Standard SKP Ordered Set** contains information following the SKP-END Symbol that is based on the LTSSM state and the sequence of Blocks. When in the Polling.Compliance state, the Symbols contain the Lane's error status information (see § Section 4.2.7.2.2 for more information). Otherwise, the Symbols contain the Lane's scrambling LFSR value, and a Data Parity bit when the SKP Ordered Set follows a Data Block. The **Control SKP Ordered Set** contains three Data Parity bits and additional information following the SKP-END-CTL Symbol.

When operating at 8.0 GT/s in Non-Flit Mode, the Data Parity bit of the Standard SKP Ordered Set is the even parity of the payload of all Data Blocks communicated by a Lane and is computed for each Lane independently¹⁰². Upstream and Downstream Port Transmitters compute the parity as follows:

- Parity is initialized when an SDS Ordered Set is transmitted.
- Parity is updated with each bit of a Data Block's payload after scrambling has been performed.
- The Data Parity bit of a Standard SKP Ordered Set transmitted immediately following a Data Block is set to the current parity.
- Parity is initialized after a Standard SKP Ordered Set is transmitted.

Upstream and Downstream Port Receivers compute and act on the parity as follows:

- Parity is initialized when an SDS Ordered Set is received.
- Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed.

¹⁰². The requirements for 8.0 GT/s operation documented here are identical to those in [PCIe-3.1].

Errata: Base 6.3
B825[△]~~◀▶~~

- When a Standard SKP Ordered Set is received immediately following a Data Block, each Lane compares the received Data Parity bit to its calculated parity. If a mismatch is detected, the Receiver must set the bit of the Port's Lane Error Status register that corresponds to the Lane's ~~↓↑default Lane number.~~ ~~↑↑Physical Lane Number.~~ The mismatch is not a Receiver Error and must not cause a Link retrain.
- Parity is initialized when a Standard SKP Ordered Set is received.

When operating at a data rate of 16.0 or 32.0 GT/s in Non-Flit Mode or 8.0, 16.0 or 32.0 GT/s in Flit Mode, the Data Parity bits of both the Standard SKP Ordered Set in ~~↓↑non-Flit Mode.~~ ~~↑↑non-Flit Mode.~~ and the Control SKP Ordered Set is the even parity of the payload of all Data Blocks communicated by a Lane and is computed for each Lane independently. Upstream and Downstream Port Transmitters compute the parity as follows:

- Parity is initialized when the LTSSM is in Recovery Speed.
- Parity is initialized when an SDS Ordered Set is transmitted.
- Parity is updated with each bit of a Data Block's payload after scrambling has been performed.
- Thus, in Flit Mode, Parity is computed post FEC.
- The Data Parity bit of a Standard SKP Ordered Set transmitted immediately following a Data Block is set to the current parity.
- The Data Parity, First Retimer Data Parity, and Second Retimer Data Parity bits of a Control SKP Ordered Set are all set to the current parity.
- Parity is initialized after a Control SKP Ordered Set is transmitted. However, parity is not initialized when a Standard SKP Ordered Set is transmitted.

Upstream and Downstream Port Receivers compute and act on the parity as follows:

- Parity is initialized when the LTSSM is in Recovery Speed.
- Parity is initialized when an SDS Ordered Set is received.
- Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed.
- Thus, in Flit Mode, Parity is compared prior to FEC decoding.
- When a Control SKP Ordered Set is received, each Lane compares the received Data Parity bit to its calculated parity. It is strongly recommended that this check be performed for Control SKP Ordered Sets immediately following a Data Block only. If a mismatch is detected, the Receiver must set the bit of the Port's 16.0 GT/s Local Data Parity Mismatch Status Register that corresponds to the Lane's ~~↓↑default Lane number.~~ ~~↑↑Physical Lane Number.~~ The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a Control SKP Ordered Set is received, each Lane compares the received First Retimer Data Parity bit to its calculated parity. It is strongly recommended that this check be performed for Control SKP Ordered Sets immediately following a Data Block only. If a mismatch is detected, the Receiver must set the bit of the Port's 16.0 GT/s First Retimer Data Parity Mismatch Status Register that corresponds to the Lane's ~~↓↑default Lane number.~~ ~~↑↑Physical Lane Number.~~ The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a Control SKP Ordered Set is received, each Lane compares the received Second Retimer Data Parity bit to its calculated parity. It is strongly recommended that this check be performed for Control SKP Ordered Sets immediately following a Data Block only. If a mismatch is detected, the Receiver must set the bit of the Port's 16.0 GT/s Second Retimer Data Parity Mismatch Status Register ~~↓↑that corresponds to the Lane's default Lane number.~~ ~~↑↑Physical Lane Number.~~ The mismatch is not a Receiver Error and must not cause a Link retrain.

Errata: Base 6.3
B825[△]~~◀▶~~

Errata: Base 6.3
B825[△]~~◀▶~~

Errata: Base 6.3
B825[△]~~◀▶~~

- When a Standard SKP Ordered Set is received immediately following a Data Block, the Receiver is permitted to compare the received Data Parity bit to its calculated parity bit. However, the results of such a comparison must not affect the state of the Lane Error Status Register .
- Parity is initialized when a Control SKP Ordered Set is received. However, parity is not initialized when a Standard SKP Ordered Set is received.

See § Section 4.3.7.7 for the definition of First Retimer and Second Retimer, and for Retimer Pseudo Port requirements for parity computation and modification of the First Retimer Data Parity and Second Retimer Data Parity bits of Control SKP Ordered Sets.

IMPLEMENTATION NOTE: LFSR IN STANDARD SKP ORDERED SET §

The LFSR value is transmitted to enable trace tools to be able to function even if they need to reacquire block alignment in the midst of a bit stream. Since trace tools cannot force the link to enter Recovery, they can reacquire bit lock, if needed, and monitor for the SKP Ordered Set to obtain Block alignment and perform Lane-to-Lane de-skew. The LFSR value from the SKP Ordered Set can be loaded into its LFSR to start interpreting the bit stream. It must be noted that with a bit stream one may alias to a SKP Ordered Set on a non-Block boundary. The trace tools can validate their Block alignment by using implementation specific means such as receiving a fixed number of valid packets or Sync Headers or subsequent SKP Ordered Sets .

Table 4-64 Standard SKP Ordered Set with 128b/130b Encoding §

| Symbol Number | Value | Description |
|--|--|--|
| 0 through (4*N-1) [N can be 1 through 5] | AAh for 8.0 GT/s and 16.0 GT/s data rates 99h for 32.0 GT/s and higher data rates | SKP Symbol. Symbol 0 is the <u>SKP Ordered Set identifier</u> . |
| 4*N | E1h | SKP_END Symbol Signifies the end of the <u>SKP Ordered Set</u> after three more Symbols. |
| 4*N + 1 | 00-FFh | (i) If LTSSM state is <u>Polling.Compliance</u> : AAh (ii) Else if prior block was a Data Block: Bit[7] = Data Parity Bit[6:0] = LFSR[22:16] (iii) Else: Bit[7] = ~LFSR[22] Bit[6:0] = LFSR[22:16] |
| 4*N + 2 | 00-FFh | (i) If the LTSSM state is <u>Polling.Compliance</u> : Error_Status[7:0] (ii) Else LFSR[15:8] |

| Symbol Number | Value | Description |
|---------------|--------|--|
| $4^*N + 3$ | 00-FFh | (i) If the LTSSM state is <u>Polling.Compliance</u> : ~Error_Status[7:0] (ii) Else: LFSR[7:0] |

The Control SKP Ordered Set is different from the Standard SKP Ordered Set in the last 4 Symbols. It is used to communicate the parity bits, as computed by each Retimer, in addition to the Data Parity bit computed by the Upstream/ Downstream Port. It may also be used for Lane Margining at a Retimer's Receiver, as described below.

Table 4-65 Control SKP Ordered Set with 128b/130b Encoding §

| Symbol Number | Value | Description |
|--|---|---|
| 0 through (4^*N-1) [N can be 1 through 5] | AAh for 8.0 GT/s and 16.0 GT/s data rates | SKP Symbol. Symbol 0 is the <u>SKP Ordered Set identifier</u> . |
| | 99h for 32.0 GT/s and higher data rates | |
| 4^*N | 78h | SKP_END_CTL Symbol. Signifies the end of the <u>Control SKP Ordered Set</u> after three more Symbols. |
| $4^*N + 1$ | 00-FFh | ↑↓ Bit 7: Data Parity Bit 6: First Retimer Data Parity Bit 5: Second Retimer Data Parity ↓ Bits ↑↓[4:0]:↑↑[7:0]: Refer to ↑↑ Margin CRC↑ ↑↑\$ Section 4.2.18.1↑↑[4:0]↓ |
| $4^*N + 2$ | 00-FFh | Bit 7: Margin Parity Bits [6:0]: Refer to § Section 4.2.18.1 |
| $4^*N + 3$ | 00-FFh | Bits [7:0]: Refer to § Section 4.2.18.1 |

ECN: Base 6.3
Optical ▲↔

The ‘Margin CRC[4:0]’ is computed from Bits [6:0] in Symbols $4N+2$ (referred to as $d[6:0]$ in the equations below, where $d[0]$ is Bit 0 of Symbol $4N+2$, $d[1]$ is Bit 1 of Symbol $4N+2$, ... $d[6]$ is Bit 6 of Symbol $4N+2$) and Bits [7:0] of Symbol $4N+3$ (referred to as $d[14:7]$, where $d[7]$ is Bit 0 of Symbol $4N+3$, $d[8]$ is Bit 1 of Symbol $4N+3$, .. $d[14]$ is Bit 7 of Symbol $4N+3$) as follows:

$$\begin{aligned} \text{Margin CRC}[0] &= d[0] \wedge d[3] \wedge d[5] \wedge d[6] \wedge d[9] \wedge d[10] \wedge d[11] \wedge d[12] \wedge d[13] \\ \text{Margin CRC}[1] &= d[1] \wedge d[4] \wedge d[6] \wedge d[7] \wedge d[10] \wedge d[11] \wedge d[12] \wedge d[13] \wedge d[14] \\ \text{Margin CRC}[2] &= d[0] \wedge d[2] \wedge d[3] \wedge d[6] \wedge d[7] \wedge d[8] \wedge d[9] \wedge d[10] \wedge d[14] \\ \text{Margin CRC}[3] &= d[1] \wedge d[3] \wedge d[4] \wedge d[7] \wedge d[8] \wedge d[9] \wedge d[10] \wedge d[11] \\ \text{Margin CRC}[4] &= d[2] \wedge d[4] \wedge d[5] \wedge d[8] \wedge d[9] \wedge d[10] \wedge d[11] \wedge d[12] \end{aligned}$$

‘Margin Parity’ is the even parity of Bits [4:0] of Symbol $4N+1$, Bits [6:0] of Symbol $4N+2$, and Bits [7:0] of Symbol $4N+3$ (i.e., Margin Parity = Margin CRC[0] ^ Margin CRC[1] ^ Margin CRC[2] ^ Margin CRC[3] ^ Margin CRC[4] ^ d[0] ^ d[1] ^ d[2] ^ d[3] ^ d[4] ^ d[5] ^ d[6] ^ d[7] ^ d[8] ^ d[9] ^ d[10] ^ d[11] ^ d[12] ^ d[13] ^ d[14]).

‘Margin Parity’ only applies to 128b/130b encoding. 1b/1b Control SKP Ordered Sets do not contain a ‘Margin Parity’ field.

The rules for generating and checking the Margin CRC and Margin Parity are described in § [Section 4.2.1.3](#).

IMPLEMENTATION NOTE: ERROR PROTECTION IN CONTROL SKP ORDERED SET §

The 21 bits in Symbol 4N+1 (Bits [4:0]), Symbol 4N+2 (Bits[7:0]) and Symbol 4N+3 (Bits[7:0]) is protected by 5 bits of CRC and one bit of parity, leaving 15 bits for information passing. The parity bit provides detection against odd number of bit-flips (e.g., 1-bit, 3-bit), whereas the CRC provides guaranteed detection of 1-bit and 2-bit flips; thus resulting in a triple bit flip detection guarantee over the 21 bits as well as guaranteed detection of burst errors of length 5. The 5-bit CRC is derived from the primitive polynomial: $x^5 + x^2 + 1$.

Since these 21 bits are not part of a TLP, repeated delivery of the same content provides delivery guarantee. This is achieved through architected registers. Downstream commands are sent from the Downstream Port, reflecting the contents of an architected register whereas the upstream status that passes the error checking is updated into a status register in the Downstream Port. Software thus has a mechanism to issue a command and wait for the status to be reflected back before issuing a new command. Thus, these 15 bits of information act as a micro-packet.

[4.2.8.3 SKP Ordered Set for 1b/1b Encoding](#) §

- Only [Control SKP Ordered Sets](#) are transmitted in 1b/1b Encoding.
- A transmitted [SKP Ordered Set](#) is 40 symbols (40B), and a received [SKP Ordered Set](#) can be 24, 32, 40, 48 or 56 symbols (24B, 32B, 40B, 48B or 56B).
- The transmitted [SKP Ordered Set](#) consists of the SKPs ([F00F_F00F](#)), followed by a [SKP-END](#) ([FFF0_00F0](#)), followed by [PHY Payload](#) (8B), as shown in § [Table 4-66](#).
- The [PHY Payload](#) field of the 1b/1b [SKP Ordered Set](#) is illustrated in § [Table 4-67](#), § [Figure 4-76](#), § [Figure 4-77](#), and § [Figure 4-78](#).
- The [Payload Type](#) and [Parity](#) fields are replicated four times, spaced more than 16 bits apart.
 - The Receiver can implement a majority voting to correct these fields if three or more sets match. If two sets of value are 0b and two sets are 1b, there is a tie. A tie in the parity can be considered to mismatch. A tie in the [Payload Type](#) results in ignoring the [PHY payload](#) field.
- The [Payload](#) field is replicated twice.
- For Margin payload with its own CRC, the Receiver can choose whichever copy passes CRC (and if both pass CRC, can perform a comparison before acting on it).

The Data Parity bit is the even parity of the payload of all Data Blocks communicated by a Lane and is computed for each Lane independently. Upstream and Downstream Port Transmitters compute the parity as follows:

- Parity is initialized when the [LTSSM](#) is in [Recovery.Speed](#).
- Parity is initialized when an [SDS Ordered Set](#) is transmitted.
- Parity is initialized after a [SKP Ordered Set](#) is transmitted.
- Parity is updated with each bit of a Data Block's payload after scrambling has been performed. Thus, gray coding and precoding, if any, does not affect the Parity calculation and Parity is computed post FEC.
- The Data Parity bit of a [SKP Ordered Set](#) transmitted immediately following a Data Block is set to the current parity.

- The Data Parity , First Retimer Data Parity , and Second Retimer Data Parity bits of a SKP Ordered Set are all set to the current parity.

Upstream and Downstream Port Receivers compute and act on the parity as follows:

- Parity is initialized when the LTSSM is in Recovery.Speed .
- Parity is initialized when an SDS Ordered Set is received.
- Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed. Thus, this is performed after gray coding and precoding, if any, by the Receiver and Parity is compared pre FEC.
- When a SKP Ordered Set is received, each Lane compares the received Data Parity bit to its calculated parity. It is strongly recommended that this check be performed for Control SKP Ordered Sets immediately following a Data Block only. If a mismatch is detected, the Receiver must set the bit of the Port's 16.0 GT/s Local Data Parity Mismatch Status Register that corresponds to the Lane's ~~↓↓default Lane number.↓↑Physical Lane Number .↑~~ The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a SKP Ordered Set is received, each Lane compares the received First Retimer Data Parity bit to its calculated parity. It is strongly recommended that this check be performed for Control SKP Ordered Sets immediately following a Data Block only. If a mismatch is detected, the Receiver must set the bit of the Port's 16.0 GT/s First Retimer Data Parity Mismatch Status Register that corresponds to the Lane's ~~↓↓default Lane number.↓↑Physical Lane Number .↑~~ The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a SKP Ordered Set is received, each Lane compares the received Second Retimer Data Parity bit to its calculated parity. It is strongly recommended that this check be performed for Control SKP Ordered Sets immediately following a Data Block only. If a mismatch is detected, the Receiver must set the bit of the Port's 16.0 GT/s Second Retimer Data Parity Mismatch Status Register that corresponds to the Lane's ~~↓↓default Lane number.↓↑Physical Lane Number .↑~~ The mismatch is not a Receiver Error and must not cause a Link retrain.
- Parity is initialized when a SKP Ordered Set is received. See § Section 4.3.7.7 for the definition of First Retimer and Second Retimer, and for Retimer Pseudo Port requirements for parity computation and modification of the First Retimer Data Parity and Second Retimer Data Parity bits of Control SKP Ordered Sets .

Errata: Base 6.3
B825△◀▷

Errata: Base 6.3
B825△◀▷

Errata: Base 6.3
B825△◀▷

Table 4-66 Control SKP Ordered Set with 1b/1b Encoding §

| Symbol Number | Value | Description |
|----------------------|-------|-------------|
| 0, 4, 8, 12, 16, 20 | F0h | SKP Symbol |
| 1, 5, 9, 13, 17, 21 | 0Fh | |
| 2, 6, 10, 14, 18, 22 | F0h | |
| 3, 7, 11, 15, 19, 23 | 0Fh | |
| 24, 28 | FFh | SKP_END |
| 25, 29 | F0h | |
| 26, 30 | 00h | |
| 27, 31 | F0h | |

Base 6.4 vs Base 6.3

| Symbol Number | Value | Description |
|---------------|---|-------------|
| 32 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[7:0] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | PHY Payload |
| 33 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[15:8] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |
| 34 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[23:16] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |
| 35 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[31:24] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |
| 36 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[39:32] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |
| 37 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[47:40] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |
| 38 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[55:48] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |
| 39 | F0h if TS0 Ordered Sets are being transmitted, as defined in § Section 4.2.7.4.2 , else Phy Payload[63:56] See § Table 4-67 , § Figure 4-76 , § Figure 4-77 , and § Figure 4-78 . | |

Base 6.4 vs Base 6.3

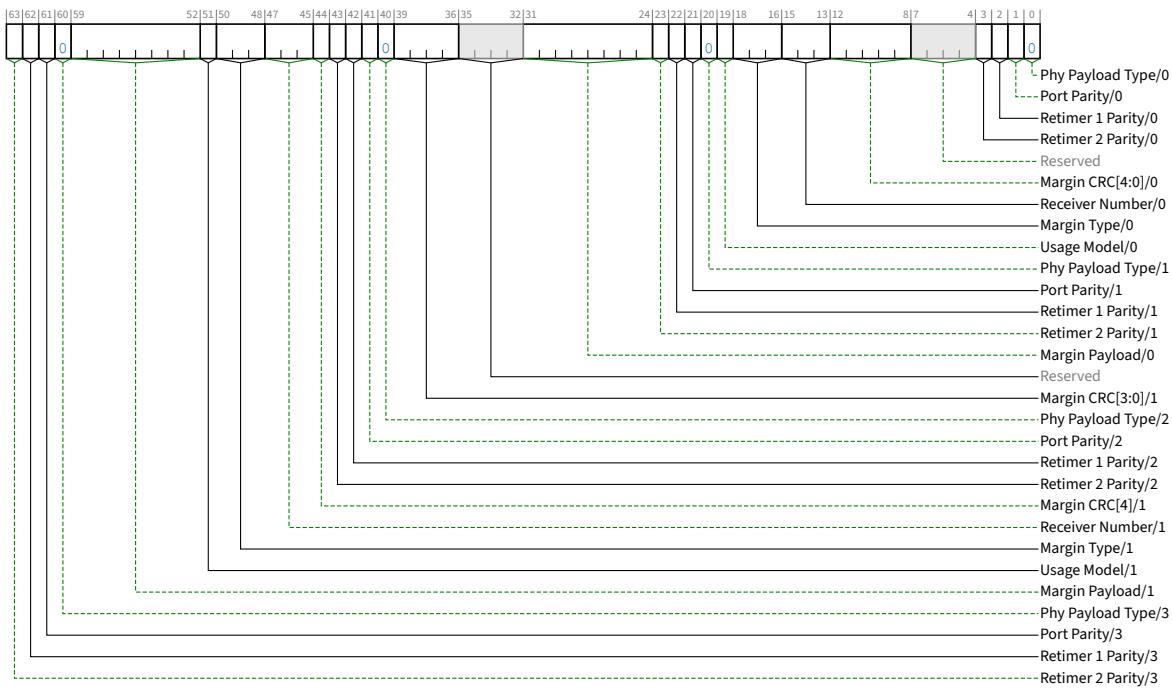


Figure 4-76 Margin PHY Payload for Control SKP Ordered Set with 1b/1b Encoding §

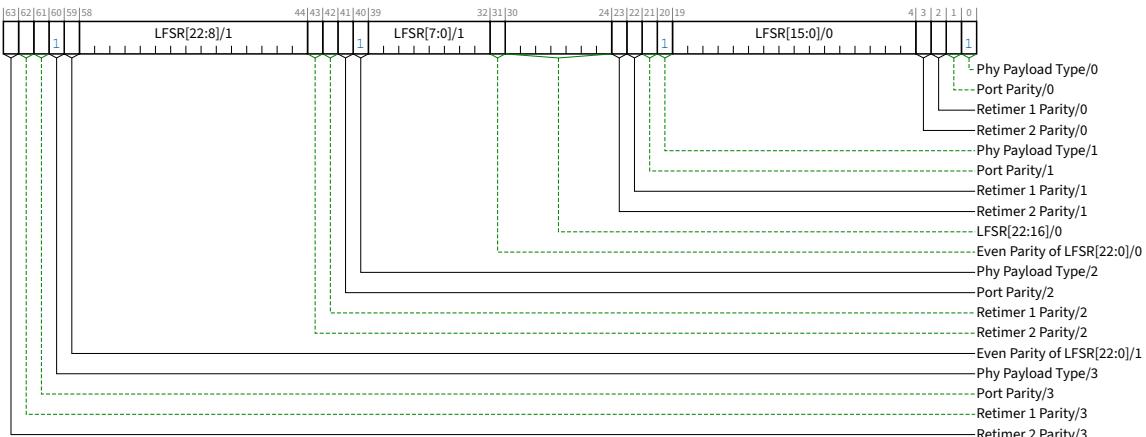


Figure 4-77 LFSR PHY Payload for Control SKP Ordered Set with 1b/1b Encoding §

Base 6.4 vs Base 6.3

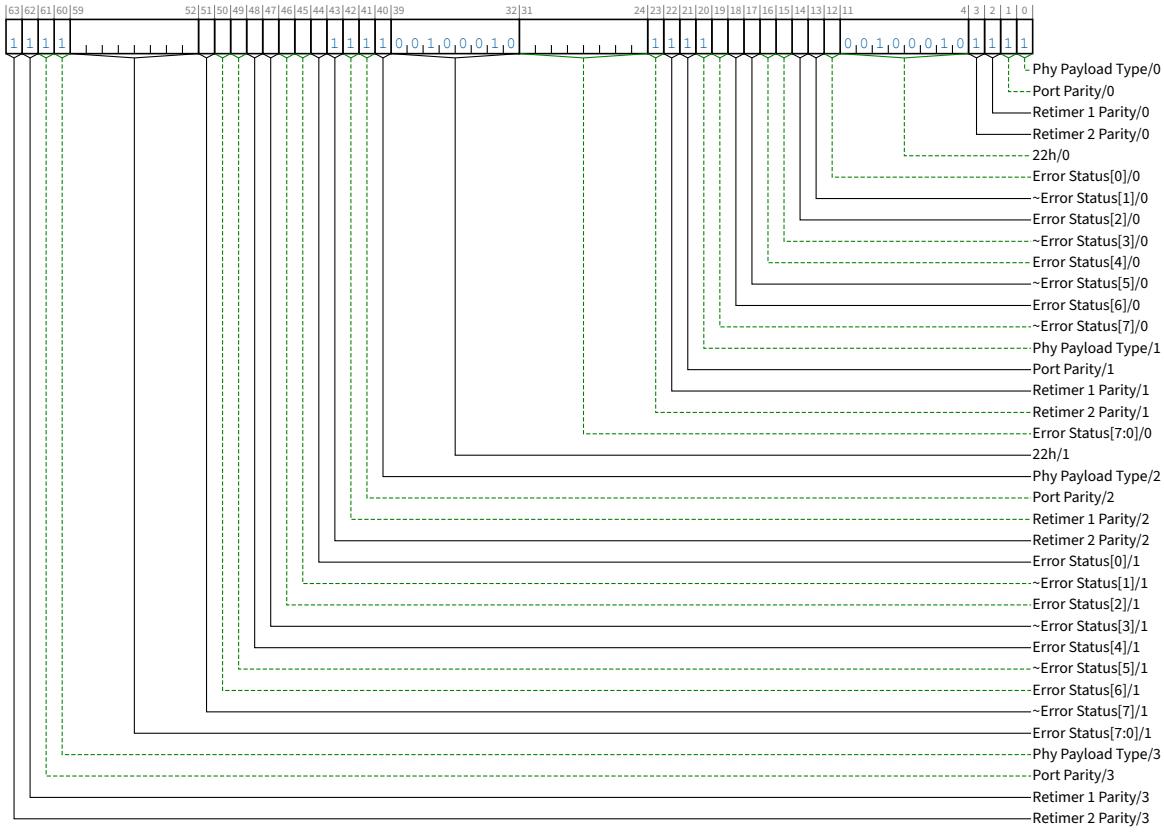


Figure 4-78 Polling.Compliance PHY Payload for Control SKP Ordered Set with 1b/1b Encoding §

Table 4-67 PHY Payload for Control SKP Ordered Set with 1b/1b Encoding §

| Field | Bit Position(s) (Replicated Bit Position(s)) | Description |
|------------------|---|--|
| Phy Payload Type | 0 (20) (40) (60) | If the LTSSM state is Polling.Compliance : this bit is Set to 1b Else: 0b Margin 1b LFSR See § Section 4.2.8.4 for Phy Payload Type field usage. |
| Port Parity | 1 (21) (41) (61) | If the LTSSM state is Polling.Compliance : this bit is Set to 1b Else: even parity of all the bits in the Data Blocks on that Lane from the last SKP Ordered Set |
| Retimer 1 Parity | 2 (22) (42) (62) | If the LTSSM state is Polling.Compliance : this bit is Set to 1b Else: Port sets this bit with Port Parity; First Retimer, if present, overwrites with its own calculated even parity of all the bits in the Data Blocks on that Lane from the last SKP ordered Set |

| Field | Bit Position(s) (Replicated Bit Position(s)) | Description |
|---------------------|---|--|
| Retimer 2 Parity | 3 (23) (43) (63) | If the LTSSM state is Polling.Compliance : this bit is Set to 1b Else: Port sets this bit with Port Parity; Second Retimer, if present, overwrites with its own calculated even parity of all the bits in the Data Blocks on that Lane from the last SKP Ordered Set |
| Payload [23:0] | {31:24, 19:4} ({59:44, 39:32}) | If the LTSSM state is Polling.Compliance : { Error_Status[7:0], ~Error_Status[7], Error_Status[6], ~Error_Status[5], Error_Status[4], ~Error_Status[3], Error_Status[2], ~Error_Status[1], Error_Status[0], 22h } Else If PHY Payload Type == 0b Margin: { Margin Payload[7:0], ↓↓Usage Model, ↓↑0b (Usage Model), ↑ Margin Type[2:0], Receiver Number[2:0], Margin CRC[4:0], Reserved[3:0] } ↓↓Else If PHY Payload Type == 0b Sideband: ↓ ↓↓ { Sideband Payload[7:0], 1b (Usage Model), ↑ ↓↓ Sideband Type, Reserved[4:0], ↑ ↓↓ Sideband CRC[4:0], Reserved[3:0]]↑ ↓↓ Sideband Type: 0b - Cmd, 1b - Ack↑ Else: { Even Parity of LFSR[22:0], LFSR[22:0] } All of these are in little-endian format. Thus, for example, the 8 bits of Margin Payload occupies bits 31:24 and 59:52 ↓↓Sideband CRC is generated using the same formula and bit positions used for Margin CRC.↑ |

ECN: Base 6.3
Optical△↔Errata: Base 6.3
B845△↔

4.2.8.4 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, all Lanes of a multi-Lane Link must simultaneously transmit SKP Ordered Sets of the same length and type (Control or Standard), except as allowed for ~~↓↓av~~ ~~↓↓in the following cases:~~
 - ~~↓↓A~~ Loopback Follower in the Loopback.Active LTSSM State (see § Section 4.2.5.11 and § Table 8-7 for the definition of simultaneous in this context).
 - ~~↓↓During L0p upsize, where lanes being activated will be sending Standard SKP Ordered Sets and lanes that are already active may be alternating between Standard and Control SKP Ordered Sets.~~
- The transmitted SKP Ordered Set when using 8b/10b encoding must follow the definition in § Section 4.2.8.1 .
- The transmitted SKP Ordered Set when using 128b/130b encoding must follow the definition in § Section 4.2.8.2 .
- The transmitted SKP Ordered Set when using 1b/1b-encoding must follow the definition in § Section 4.2.8.3 .
- When using 8b/10b encoding:
 - If the Link is not operating in SRIS , or the bit corresponding to the current Link speed is Set in the Enable Lower SKP OS Generation Vector field and the LTSSM is in L0, a SKP Ordered Set must be scheduled for transmission at an interval between 1180 and 1538 Symbol Times.

- If the Link is operating in SRIS and either the bit corresponding to the current Link speed is Clear in the Enable Lower SKP OS Generation Vector field or the LTSSM is not in L0, a SKP Ordered Set must be scheduled for transmission at an interval of less than 154 Symbol Times.
- When using 128b/130b encoding:
 - If the Link is not operating in SRIS, or the bit corresponding to the current Link speed is Set in the Enable Lower SKP OS Generation Vector field and the LTSSM is in L0, a SKP Ordered Set must be scheduled for transmission at an interval between 370 and 375 Blocks, in Non-Flit Mode or while not sending Data Stream in Flit Mode. Loopback Followers must meet this requirement until they start looping back the incoming bit stream.
 - If the Link is operating in SRIS and either the bit corresponding to the current Link speed is Clear in the Enable Lower SKP OS Generation Vector field or the LTSSM is not in L0, a SKP Ordered Set must be scheduled for transmission at an interval less than 38 Blocks, in Non-Flit Mode or while not sending Data Stream in Flit Mode. Loopback Followers must meet this requirement until they start looping back the incoming bit stream.
 - When the LTSSM is in the Loopback state and the Link is not operating in SRIS, the Loopback Lead must schedule two SKP Ordered Sets to be transmitted, at most two Blocks apart from each other, at an interval between 370 to 375 blocks. For 32.0 GT/s, the Loopback Lead is permitted to have an EIEOSQ between the two SKP Ordered Sets.
 - When the LTSSM is in the Loopback state and the Link is operating in SRIS, the Loopback Lead must schedule two SKP Ordered Sets to be transmitted, at most two Blocks apart from each other, at an interval of less than 38 Blocks. For 32.0 GT/s, the Loopback Lead is permitted to have an EIEOSQ between the two SKP Ordered Sets.
 - SKP Ordered Set transmission rules are specified in § Section 4.2.3.4.6 when all the following conditions are true:
 - The Link is operating in Flit Mode.
 - The LTSSM is not in the Loopback state.
 - The transmitter is either transmitting a Data Stream or is transmitting a SKPOS just prior to a Data Stream.
 - The Control SKP Ordered Set is transmitted only at the following times:
 - In Non-Flit Mode when the data rate is 16.0 or 32.0 GT/s and in Flit Mode when the data rate is 8.0, 16.0, or 32.0 GT/s and one of the following conditions is met:
 - A Data Stream is being transmitted.
SKP Ordered Sets transmitted within a Data Stream must alternate between the Standard SKP Ordered Set and the Control SKP Ordered Set except for when undergoing an L0p upsize. In L0p when an inactive Lane is transitioning to active the Link must send a Control SKP Ordered Set on all Lanes prior to transmitting the Data Stream on the transitioning Lanes. During an L0p upsize, active lanes are permitted to transmit Control SKP Ordered Sets instead of Standard SKP Ordered Sets to reduce the latency of the that L0p upsize.
 - The LTSSM enters the Configuration.Idle state or Recovery.Idle state.
See § Section 4.2.7.3.6 and § Section 4.2.7.4.5 for more information. Transmission of this instance of the Control SKP Ordered Set is not subject to any minimum scheduling interval requirements described above. Transmitters are permitted, but not required, to reset their SKP Ordered Set scheduling interval timer after transmitting this instance of the Control SKP Ordered Set.
 - The first SKP Ordered Set is being sent after an L0p upsizing.

A Control SKP Ordered Set must be transmitted on all Lanes - the Lanes that are active and the Lanes that are being activated.

- ↑↑It is strongly recommended that Lanes being activated during L0p upsize send Standard SKP Ordered sets while in an Ordered Set stream. However, those lanes are permitted to send the same SKP Ordered Set type being sent on active Lanes.↑
 - ↑↑If Control SKP Ordered Sets are sent on Lanes being activated during L0p upsize, the Data Parity bits of the Control SKP Ordered Sets are undefined, and Receivers are strongly recommended to ignore the received Data Parity bits on the Lanes being activated.↑
- When using 1b/1b encoding: only Control SKP Ordered Sets are sent. The following rules must be followed, except when transmitting Compliance Modified Compliance patterns:
 - If the Link is not operating in SRIS, or the bit corresponding to the current Link speed is Set in the Enable Lower SKP OS Generation Vector field and the LTSSM is in L0, a SKP Ordered Set must be scheduled for transmission at an interval of 740 to 750 Blocks from the prior SKP Ordered Set, while a Data Stream is not in progress. Loopback Followers must meet this requirement until they start looping back the incoming bit stream.
 - Behavior is undefined if the Enable Lower SKP OS Generation Vector field setting is changed in FM at a data rate other than 2.5 GT/s.
 - If the Link is operating in SRIS and either the bit corresponding to the current Link speed is Clear in the Enable Lower SKP OS Generation Vector field or the LTSSM is not in L0, a SKP Ordered Set must be scheduled for transmission at an interval of less than 76 Blocks, while a Data Stream is not in progress. Loopback Followers must meet this requirement until they start looping back the incoming bit stream.
 - When the LTSSM is in the Loopback state and the Link is not operating in SRIS :
 - If the Loopback Lead is operating in a Data Stream:
 - A single SKP Ordered Set is sent at the interval specified in § Table 4-32 .
 - If the Loopback Lead is not operating in a Data Stream:
 - A single SKP Ordered Set must be sent at an interval of 740 to 750 blocks from the prior SKP Ordered Set .
 - Optionally, two back-to-back SKP Ordered Sets are allowed to be scheduled for transmission when operating at 64.0 GT/s.
 - When the LTSSM is in the Loopback state and the Link is operating in SRIS :
 - If the Loopback Lead is operating in a Data Stream:
 - A single SKP Ordered Set is sent at the interval specified in § Table 4-32 .
 - If the Loopback Lead is not operating in a Data Stream:
 - A single SKP Ordered Set must be sent at an interval of less than 76 Blocks.
 - Optionally, two back-to-back SKP Ordered Sets are allowed to be scheduled for transmission when operating at 64.0 GT/s.
 - During the Data Stream, just prior to transitioning to transmitting the Data Stream, or immediately after terminating the Data Stream, SKP Ordered Set transmission rules must be followed as specified in § Section 4.2.3.1.5 .
 - Phy Payload Type must be set as follows:
 - If the LTSSM state is Polling.Compliance : 1b.

Errata: Base 6.3
B845△◀

- Else: For Lane 0, alternate between 0b and 1b in consecutive Control SKP OS, with any starting value after an exit from electrical idle. Any other Lane must send the same value as Lane 0. Receivers must not depend on this alternation.
- Scheduled SKP Ordered Sets shall be transmitted if a packet or Ordered Set is not already in progress, otherwise they are accumulated and then inserted consecutively at the next packet or Ordered Set boundary. Note: When operating in Flit Mode at 128b/130b encoding and not in the Loopback LTSSM state, or 1b/1b encoding in any LTSSM state, SKP Ordered Sets cannot be transmitted in consecutive Blocks within a Data Stream. See § Section 4.2.3.4.6 for more information.
- SKP Ordered Sets do not count as an interruption when monitoring for consecutive Symbols or Ordered Sets (e.g., eight consecutive TS1 Ordered Sets in Polling.Active).
- When using 8b/10b encoding: SKP Ordered Sets must not be transmitted while the Compliance Pattern or the Modified Compliance Pattern (see § Section 4.2.9) is in progress during Polling.Compliance if the Compliance SOS bit of the Link Control 2 register is 0b. If the Compliance SOS bit of the Link Control 2 register is 1b, two consecutive SKP Ordered Sets must be sent (instead of one) for every scheduled SKP Ordered Set time interval while the Compliance Pattern or the Modified Compliance Pattern is in progress when using 8b/10b encoding.
- When using 128b/130b or 1b/1b encoding: The Compliance SOS register bit has no effect. While in Polling.Compliance , Transmitters must not transmit any SKP Ordered Sets other than those specified as part of the Modified Compliance Pattern in § Section 4.2.12 .
- Any and all time spent in a state when the Transmitter is electrically idle does not count in the scheduling interval used to schedule the transmission of SKP Ordered Sets .
- It is recommended that any counter(s) or other mechanisms used to schedule SKP Ordered Sets be reset any time when the Transmitter is electrically idle.

IMPLEMENTATION NOTE: SKP OS IN FLIT MODE 1B/1B ENCODING §

While operating in Flit Mode at 1b/1b encoding, SKP Ordered Sets cannot be sent in consecutive Blocks within a Data Stream because Receivers expect Ordered Sets to be sent only at specific Flit boundaries. See § Table 4-32 for more information.

4.2.8.5 Rules for Receivers §

- Receivers shall recognize received SKP Ordered Sets as defined in § Section 4.2.8.1 when using 8b/10b encoding, as defined in § Section 4.2.8.2 when using 128b/130b encoding, and as defined in § Section 4.2.8.3 when using the 1b/1b encoding.
 - The length of the received SKP Ordered Sets shall not vary from Lane-to-Lane in a multi-Lane Link, except as may occur during Loopback.Active .
- Receivers must be tolerant to receive and process SKP Ordered Sets sent by a transmitter, as defined by the appropriate rules in [11↑§ Section 4.2.3.4.6](#) , § Section 4.2.8.1 , § Section 4.2.8.2 , and § Section 4.2.8.3
 - Note: Since Transmitters in electrical idle are not required to reset their mechanism for time-based scheduling of SKP Ordered Sets , Receivers shall be tolerant to receive the first time-scheduled SKP Ordered Set following electrical idle in less than the average time interval between SKP Ordered Sets .
- For 8.0, 16.0, and 32.0 GT/s data rates, in L0 state, Receivers must check that each SKP Ordered Set is preceded by a Data Block with an EDS token.

- Receivers shall be tolerant to receive and process consecutive SKP Ordered Sets in 2.5 GT/s and 5.0 GT/s data rates.
 - Receivers shall be tolerant to receive and process SKP Ordered Sets that have a maximum separation dependent on the largest Rx_MPS_Limit a component supports. For 2.5 GT/s and 5.0 GT/s data rates, the formula for the maximum number of Symbols (N) between SKP Ordered Sets is:

$$N = 1538 + \text{Rx_MPS_Limit (in bytes)} + 28.$$
 For example, if Rx_MPS_Limit is 4096 bytes, N = 1538 + 4096 + 28 = 5662.
- When using 1b/1b encoding, each Receiver is permitted to insert or delete 8 Bytes of SKP at an aligned 8 byte boundary. Thus, a received SKP Ordered Set can be 24, 32, 40, 48 or 56 bytes.

IMPLEMENTATION NOTE: SKP ORDERED SETS IN A DATA STREAM IN FLIT MODE §

- For 1b/1b, Receivers can predict when SKP OS will occur.
- For 128b/130b, Receivers can predict when SKP OS will occur. This is needed since the EDS Token is not used in Flit Mode.
- For 8b/10b, Receivers need to look for COM SKP. SKP OS will occur at a period specified in § Section 4.2.8.4.

When upconfiguring Lanes in L0p :

- For 8b/10b, there is no CTL SKP OS but there also is no retimer parity bits that need to be initialized.
- For 128b/130b, the CTL SKP OS is needed to restart the parity computations
- For 1b/1b, CTL SKP OS are always sent to restart the parity computations. Identical values in Phy Payload type are required for simplicity (see § Section 4.2.8.4)

4.2.8.6 ↑↑SKP Ordered Set Usage with Optical Retimer Solutions↑ §

ECN: Base 6.3
Optical△◀▷

↑↑The following rules apply (in addition to the other requirements described in Section 4.2.8) when an Optical Retimer Solution is present. The presence of an Optical Retimer Solution is indicated when the Optical_RT_present variable is set to 1b.↑

- ↑↑It is strongly recommended that the Electrical Sub-Links operate in SRIS mode. SKP Ordered Sets may be transmitted at the lower SRNS rate in SRIS mode if all necessary conditions are met (see Section 4.2.8). Determining that the Optical Retimer Solution supports the SRNS rate is implementation specific. Shared reference clocking (common RefClk) is also permitted in an implementation specific manner when a Optical Retimer Solution is present.↑
- ↑↑All components on the Link are permitted to use the SKP Ordered Sets to communicate sideband information when operating at 8.0 GT/s or higher (see Table 4-55 and Table 4-62). When a Sideband Payload is transmitted, the Sideband Type field must be set to 0b, indicating a command. It must continue to be sent until an acknowledgement is received in the form of a SKP Ordered Set with a Sideband Payload matching the transmitted Sideband Payload and the Sideband Type set to 1b. The usage of SKP Ordered Sets carrying sideband information is implementation specific. The Sideband Payload may target the Component at the other end of the Link, the Upstream Pseudo Port or the Downstream Pseudo Port of the optical extension component. The payload definition is outside the scope of this specification.↑

- ↑Sideband signals may optionally be communicated using the SKP Ordered Sets in sideband usage mode, in an implementation specific manner. Examples of such sideband signals are: WAKE#, CLKREQ#, PWRBRK#.
- ↑The sideband mechanism is enabled when the following conditions are met:
 - ↑The Component's Optical_RT_present variable is set to 1b. Alternatively, the presence of an Optical Retimer Solution may be determined in an implementation specific manner.
 - ↑Control SKP Ordered Sets are transmitted and received with Usage Model set to 1b, Sideband Type set to 0b, and Sideband Payload set to a predetermined initialization sideband value (value is implementation specific).
 - ↑Control SKP Ordered Sets are transmitted and received with Usage Model set to 1b, Sideband Type set to 1b, and Sideband Payload set to sideband value that is being Ack'd (predetermined value for sideband initialization).
 - ↑As an alternative to the exchange of SKP Ordered Sets as described above, components are permitted to determine support for the SKP Ordered Set sideband mechanism in an implementation specific manner.

IMPLEMENTATION NOTE:

SIDEBAND COMMUNICATION WITH 8B/10B ENCODED RATES §

↑Sideband communication using SKP Ordered Sets is not defined for 8b/10b encoding. Due to the limited availability of SKP symbols, a significant change to the format of the SKP Ordered Set would be needed, posing challenges to backwards compatibility. An implementation that deploys an Optical Retimer Solution is not prevented from either using the Training Set Messaging mechanism available in Modified TS1/TS2 Ordered Sets or repurposing one TS1 and/or TS2 identifier to exchange information across the Link prior to LinkUp.

IMPLEMENTATION NOTE:

BANDWIDTH AVAILABLE USING SKP OS SIDEBAND COMMUNICATION §

↑The calculations below illustrate the expected sideband bandwidth available when operating in SRIS mode and when operating in either Common Clock or SRNS modes.

- ↑At 64.0 GT/s – OS interval is after 4 Flits for x1 in SRIS Mode
 - ↑1B of sideband payload
 - ↑4 Flits + 1 SKP OS = 4(256B) + 40B = 1064B
 - ↑Available bandwidth = 1/1064 * 64.0 GT/s = ~60Mb/s
- ↑At 64.0 GT/s – OS interval is after 46 Flits for x1 in Common Clock/ SRNS Mode
 - ↑1B of sideband payload
 - ↑46 Flits + 1 SKP OS = 46(256B) + 40B = 11816B
 - ↑Available bandwidth = 1/11816 * 64.0 GT/s = ~5.4Mb/s

4.2.9 Compliance Pattern in 8b/10b Encoding §

During Polling, the Polling.Compliance substate must be entered from Polling.Active based on the conditions described in § Section 4.2.7.2.1 . The compliance pattern consists of the sequence of 8b/10b Symbols K28.5 , D21.5, K28.5 , and D10.2 repeating. The Compliance sequence is as follows:

| Symbol | <u>K28.5</u> | D21.5 | <u>K28.5</u> | D10.2 |
|-------------------|--------------|------------|--------------|------------|
| Current Disparity | Negative | Positive | Positive | Negative |
| Pattern | 0011111010 | 1010101010 | 1100000101 | 0101010101 |

The first Compliance sequence Symbol must have negative disparity. It is permitted to create a disparity error to align the running disparity to the negative disparity of the first Compliance sequence Symbol.

For any given device that has multiple Lanes, every eighth Lane is delayed by a total of four Symbols. A two Symbol delay occurs at both the beginning and end of the four Symbol Compliance Pattern sequence. A x1 device, or a xN device operating a Link in x1 mode, is permitted to include the Delay Symbols with the Compliance Pattern.

This delay sequence on every eighth Lane is then:

| Symbol: | D | D | <u>K28.5</u> | D21.5 | <u>K28.5</u> | D10.2 | D | D |
|---------|---|---|--------------|-------|--------------|-------|---|---|
|---------|---|---|--------------|-------|--------------|-------|---|---|

Where D is a K28.5 Symbol. The first D Symbol has negative disparity to align the delay disparity with the disparity of the Compliance sequence.

After the eight Symbols are sent, the delay Symbols are advanced to the next Lane, until the delay Symbols have been sent on all eight lanes. Then the delay Symbols cycle back to Lane 0, and the process is repeated. It is permitted to advance the delay sequence across all eight lanes, regardless of the number of lanes detected or supported. An illustration of this process is shown below:

| | | | | | | | | | | | | |
|--------|----------------|--|----------------|-------|----------------|-------|----------------|-------|----------------|-------|----------------|-------|
| Lane 0 | D | D | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | D | D | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 1 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | D | D | <u>K28.5 -</u> | D21.5 |
| Lane 2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 3 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 4 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 5 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 6 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 7 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 8 | D | D | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | D | D | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 |
| Lane 9 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | D | D | <u>K28.5 -</u> | D21.5 |
| Key: | <u>K28.5 -</u> | COM when disparity is negative, specifically: "0011111010" | | | | | | | | | | |
| | <u>K28.5 +</u> | COM when disparity is positive, specifically: "1100000101" | | | | | | | | | | |
| | <u>D21.5</u> | Out of phase data Symbol, specifically: "1010101010" | | | | | | | | | | |
| | <u>D10.2</u> | Out of phase data Symbol, specifically: "0101010101" | | | | | | | | | | |

D

Delay Symbol K28.5 (with appropriate disparity)

This sequence of delays ensures interference between adjacent Lanes, enabling measurement of the compliance pattern under close to worst-case Inter-Symbol Interference and crosstalk conditions.

4.2.10 Modified Compliance Pattern in 8b/10b Encoding §

The Modified Compliance Pattern consists of the same basic Compliance Pattern sequence (see § Section 4.2.9) with one change. Two identical error status Symbols followed by two K28.5 are appended to the basic Compliance sequence of 8b/10b Symbols (K28.5 , D21.5, K28.5 , and D10.2) to form the Modified Compliance Sequence of (K28.5 , D21.5, K28.5 , D10.2, error status Symbol, error status Symbol, K28.5 , K28.5). The first Modified Compliance Sequence Symbol must have negative disparity. It is permitted to create a disparity error to align the running disparity to the negative disparity of the first Modified Compliance Sequence Symbol. For any given device that has multiple Lanes, every eighth Lane is moved by a total of eight Symbols. Four Symbols of K28.5 occurs at the beginning and another four Symbols of K28.7 occurs at the end of the eight Symbol Modified Compliance Pattern sequence. The first D Symbol has negative disparity to align the delay disparity with the disparity of the Modified Compliance Sequence. After the 16 Symbols are sent, the delay Symbols are advanced to the next Lane, until the delay Symbols have been sent on all eight lanes. Then the delay Symbols cycle back to Lane 0, and the process is repeated. It is permitted to advance the delay sequence across all eight lanes, regardless of the number of lanes detected or supported. A x1 device, or a xN device operating a Link in x1 mode, is permitted to include the Delay symbols with the Modified Compliance Pattern.

An illustration of the Modified Compliance Pattern is shown in § Table 4-71 . Note: This table was “wrapped” to allow it to fit on the page.

Table 4-71 Illustration of Modified Compliance Pattern §

| Lane0 | D | D | D | D | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | → next row |
|-------|------------|-------|---------|---------|---------|---------|---------|---------|---------|------------|
| | prev row → | ERR | K28.5 - | K28.5 + | K28.7 - | K28.7 - | K28.7 - | K28.7 - | K28.5 - | D21.5 |
| Lane1 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | D | D |
| Lane2 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | D21.5 |
| Lane3 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | D21.5 |
| Lane4 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | D21.5 |
| Lane5 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | D21.5 |
| Lane6 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | D21.5 |
| Lane7 | K28.5 - | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | → next row |
| | prev row → | D21.5 | K28.5 + | D10.2 | ERR | ERR | K28.5 - | K28.5 + | K28.5 - | D21.5 |

| | D | D | D | D | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | ERR | → next row | | |
|-----------------------|----------------|--|--|----------------|----------------|----------------|----------------|----------------|----------------|------------|--|--|
| Lane8 | prev row → | ERR | <u>K28.5 -</u> | <u>K28.5 +</u> | <u>K28.7 -</u> | <u>K28.7 -</u> | <u>K28.7 -</u> | <u>K28.7 -</u> | <u>K28.5 -</u> | D21.5 | | |
| Lane9 | <u>K28.5 -</u> | D21.5 | <u>K28.5 +</u> | D10.2 | ERR | ERR | <u>K28.5 -</u> | <u>K28.5 +</u> | <u>K28.5 -</u> | → next row | | |
| | prev row → | D21.5 | <u>K28.5 +</u> | D10.2 | ERR | ERR | <u>K28.5 -</u> | <u>K28.5 +</u> | D | D | | |
| Key: | | <u>K28.5 -</u> | COM when disparity is negative, specifically: "0011111010" | | | | | | | | | |
| <u>K28.5 +</u> | | COM when disparity is positive, specifically: "1100000101" | | | | | | | | | | |
| <u>D21.5</u> | | Out of phase data Symbol specifically: "1010101010" | | | | | | | | | | |
| <u>D10.2</u> | | Out of phase data Symbol, specifically: "0101010101" | | | | | | | | | | |
| D | | Delay Symbol K28.5 (with appropriate disparity) | | | | | | | | | | |
| ERR | | error status Symbol (with appropriate disparity) | | | | | | | | | | |
| <u>K28.7 -</u> | | EIE when disparity is negative, specifically "001111000" | | | | | | | | | | |
| → next row | | This table was wrapped so it fits on the page. The column after → next row is the one following prev row → | | | | | | | | | | |
| prev row → | | | | | | | | | | | | |

The reason two identical error Symbols are inserted instead of one is to ensure disparity of the 8b/10b sequence is not impacted by the addition of the error status Symbol.

All other Compliance Pattern rules are identical (i.e., the rules for adding delay Symbols) so as to preserve all the crosstalk characteristics of the Compliance Pattern.

The error status Symbol is an 8b/10b data Symbol, maintained on a per-Lane basis, and defined in 8-bit domain in the following way:

- Receiver Error Count (Bits 6:0) - Incremented on every Receiver error after the Pattern Lock bit becomes asserted.
- Pattern Lock (Bit 7) - Asserted when the Lane locks to the incoming Modified Compliance Pattern.

4.2.11 Compliance Pattern in 128b/130b Encoding §

The compliance pattern consists of the following repeating sequence of 36 or 37 Blocks:

1. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of 64 1's followed by 64 0's
2. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of the following:

| | Lane No modulo 8 = 0 | Lane No modulo 8 = 1 | Lane No modulo 8 = 2 | Lane No modulo 8 = 3 | Lane No modulo 8 = 4 | Lane No modulo 8 = 5 | Lane No modulo 8 = 6 | Lane No modulo 8 = 7 |
|----------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Symbol 0 | 55h | FFh | FFh | FFh | 55h | FFh | FFh | FFh |
| Symbol 1 | 55h | FFh | FFh | FFh | 55h | FFh | FFh | FFh |
| Symbol 2 | 55h | 00h | FFh | FFh | 55h | FFh | FFh | FFh |
| Symbol 3 | 55h | 00h | FFh | FFh | 55h | FFh | F0h | F0h |
| Symbol 4 | 55h | 00h | FFh | C0h | 55h | FFh | 00h | 00h |
| Symbol 5 | 55h | 00h | C0h | 00h | 55h | E0h | 00h | 00h |

Base 6.4 vs Base 6.3

| | Lane No modulo 8 = 0 | Lane No modulo 8 = 1 | Lane No modulo 8 = 2 | Lane No modulo 8 = 3 | Lane No modulo 8 = 4 | Lane No modulo 8 = 5 | Lane No modulo 8 = 6 | Lane No modulo 8 = 7 |
|-----------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Symbol 6 | 55h | 00h | 00h | 00h | 55h | 00h | 00h | 00h |
| Symbol 7 | {P,~P} |
| Symbol 8 | 00h | 1Eh | 2Dh | 3Ch | 4Bh | 5Ah | 69h | 78h |
| Symbol 9 | 00h | 55h | 00h | 00h | 00h | 55h | 00h | F0h |
| Symbol 10 | 00h | 55h | 00h | 00h | 00h | 55h | 00h | 00h |
| Symbol 11 | 00h | 55h | 00h | 00h | 00h | 55h | 00h | 00h |
| Symbol 12 | 00h | 55h | 0Fh | 0Fh | 00h | 55h | 07h | 00h |
| Symbol 13 | 00h | 55h | FFh | FFh | 00h | 55h | FFh | 00h |
| Symbol 14 | 00h | 55h | FFh | FFh | 7Fh | 55h | FFh | 00h |
| Symbol 15 | 00h | 55h | FFh | FFh | FFh | 55h | FFh | 00h |

Key:

P Indicates the 4-bit encoding of the Transmitter preset value being used.**~P** Indicates the bit-wise inverse of P.

3. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of the following:

| | Lane No modulo 8 = 0 | Lane No modulo 8 = 1 | Lane No modulo 8 = 2 | Lane No modulo 8 = 3 | Lane No modulo 8 = 4 | Lane No modulo 8 = 5 | Lane No modulo 8 = 6 | Lane No modulo 8 = 7 |
|-----------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Symbol 0 | FFh | FFh | 55h | FFh | FFh | FFh | 55h | FFh |
| Symbol 1 | FFh | FFh | 55h | FFh | FFh | FFh | 55h | FFh |
| Symbol 2 | FFh | FFh | 55h | FFh | FFh | FFh | 55h | FFh |
| Symbol 3 | F0h | F0h | 55h | F0h | F0h | F0h | 55h | F0h |
| Symbol 4 | 00h | 00h | 55h | 00h | 00h | 00h | 55h | 00h |
| Symbol 5 | 00h | 00h | 55h | 00h | 00h | 00h | 55h | 00h |
| Symbol 6 | 00h | 00h | 55h | 00h | 00h | 00h | 55h | 00h |
| Symbol 7 | {P,~P} |
| Symbol 8 | 00h | 1Eh | 2Dh | 3Ch | 4Bh | 5Ah | 69h | 78h |
| Symbol 9 | 00h | 00h | 00h | 55h | 00h | 00h | 00h | 55h |
| Symbol 10 | 00h | 00h | 00h | 55h | 00h | 00h | 00h | 55h |
| Symbol 11 | 00h | 00h | 00h | 55h | 00h | 00h | 00h | 55h |
| Symbol 12 | FFh | 0Fh | 0Fh | 55h | 0Fh | 0Fh | 0Fh | 55h |
| Symbol 13 | FFh | FFh | FFh | 55h | FFh | FFh | FFh | 55h |

| | Lane No modulo 8 = 0 | Lane No modulo 8 = 1 | Lane No modulo 8 = 2 | Lane No modulo 8 = 3 | Lane No modulo 8 = 4 | Lane No modulo 8 = 5 | Lane No modulo 8 = 6 | Lane No modulo 8 = 7 |
|-----------|----------------------|--|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Symbol 14 | FFh | FFh | FFh | 55h | FFh | FFh | FFh | 55h |
| Symbol 15 | FFh | FFh | FFh | 55h | FFh | FFh | FFh | 55h |
| Key: | P | Indicates the 4-bit encoding of the Transmitter preset being used. | | | | | | |
| | ~P | Indicates the bit-wise inverse of P. | | | | | | |

4. One EIEOSQ
5. 32 Data Blocks, each with a payload of 16 Idle data Symbols (00h) scrambled

IMPLEMENTATION NOTE: FIRST TWO BLOCKS OF THE COMPLIANCE PATTERN §

The first block is a very low frequency pattern to help with measurement of the preset settings. The second block is to notify the Lane number and preset encoding the compliance pattern is using along with ensuring the entire compliance pattern is DC Balanced.

The payload in each Data Block is the output of the scrambler in that Lane (i.e., input data is 0b). The Lane numbers used to determine the scrambling LFSR seed value depend on how Polling.Compliance is entered. (See § Section 4.2.2.4 for more on assigning Lane numbers and other Scrambler requirements.) The Data Blocks of the compliance pattern do not form a Data Stream and hence are exempt from the requirement of transmitting an SDS Ordered Set or EDS Token during Ordered Set Block to Data Block transition and vice-versa.

IMPLEMENTATION NOTE: ORDERED SETS IN COMPLIANCE AND MODIFIED COMPLIANCE PATTERNS IN 128B/130B ENCODING §

The various Ordered Sets (e.g., EIEOS and SKP OS) follow the Ordered Set definition corresponding to the current Data Rate of operation. For example, at 32.0 GT/s Data Rate, the EIEOS is the 32.0 GT/s EIEOS ; at 16.0 GT/s Data Rate, the EIEOS is the 16.0 GT/s EIEOS ; whereas at 8.0 GT/s Data Rate, the EIEOS is the 8.0 GT/s EIEOS defined earlier. As defined in § Section 4.2.8 , the SKP Ordered Set is the Standard SKP Ordered Set .

4.2.12 Modified Compliance Pattern in 128b/130b Encoding §

The modified compliance pattern, when not operating in SRIS , consists of repeating the following sequence of 65792 or 65793 Blocks:

1. One EIEOSQ
2. 256 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled
3. 255 sets of the following sequence:
 - i. One SKP Ordered Set

- ii. 256 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled

The modified compliance pattern, when operating in SRIS, consists of repeating the following sequence of 67585 or 67586 Blocks:

1. One EIEOSQ
2. 2048 sets of the following sequence:
 - i. One SKP Ordered Set
 - ii. 32 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled

The payload in each Data Block is the output of the scrambler in that Lane (i.e., input data is 0b). The Lane numbers used to determine the scrambling LFSR seed value depend on how Polling.Compliance is entered. (See § Section 4.2.2.4 for more on assigning Lane numbers.) The Data Blocks of the modified compliance pattern do not form a Data Stream and hence are exempt from the requirement of transmitting an SDS Ordered Set or EDS Token during Ordered Set Block to Data Block transition and vice-versa.

4.2.13 Jitter Measurement Pattern in 128b/130b §

The jitter measurement pattern consists of repeating the following Block:

- Sync Header of 01b followed by a 128-bit unscrambled payload of 16 Symbols of 55h

This generates a pattern of alternating 1s and 0s for measuring the transmitter's jitter.

4.2.14 Compliance Pattern in 1b/1b Encoding §

The Compliance Pattern consists of the following repeating sequence of 137 Blocks. Gray-coding and precoding are not performed during the compliance pattern.

1. One block, unscrambled: 64 UIs of 11b each (voltage level 3 throughout)
2. One block, unscrambled: 64 UIs of 00b each (voltage level 0 throughout)
3. Two unscrambled blocks of Toggle Pattern (Ch repeated 32 times for each block)
4. Two blocks with an unscrambled payload of the following: This is in hex format starting with the most significant Symbol. For example, in Lane 1, Block 5 (**38_DA_CC_C4_E2_3F_1D_35_3B_25_63_CC_B2_CC_FF_FF**), Symbol 0 is FFh and Symbol 15 is 38. Note that these are inserted for establishing DC balance.

Lane 0, 8:

Block 5: **FF_FF_FF_FF_FF_FF_FF_FF_FF_FF_FF_FF_FF_FF_FF**
 Block 6: **5F_26_CC_65_C2_3B_C5_3F_FF_FF_FF_FF_FF_FF**

Lane 1, 9:

Block 5: **38_DA_CC_C4_E2_3F_1D_35_3B_25_63_CC_B2_CC_FF_FF**
 Block 6: **B1_CB_0D_33_C5_D4_EC_32_A2_AC_CC_53_DC_C6_38_B4**

Lane 2, 10:

Block 5: **52_3D_D4_99_EC_0F_3C_4E_56_00_00_00_00_00_00**
 Block 6: **A5_2A_69_C3_C9_C3_93_2F_30_CF_1C_C3_37_C0_D8_ED**

Lane 3, 11:

Block 5: E8_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00
 Block 6: AC_71_6C_3C_CC_93_73_52_8E_6C_DC_0C_E7_C4_E2_D0

Lane 4, 12:

Block 5: 93_6C_C8_3A_63_93_70_F6_6F_FF_FF_FF_FF_FF_FF_FF
 Block 6: 6A_65_DC_1D_A4_DD_28_F1_C3_2E_4F_2C_5C_2D_D6_C2

Lane 5, 13:

Block 5: 33_C3_9C_6D_60_CF_71_8D_C3_35_D2_8E_3C_6D_0D_DD
 Block 6: 5E_C4_8C_ED_6C_4E_53_33_6D_C2_D5_3C_33_28_FC_53

Lane 6, 14:

Block 5: 00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00
 Block 6: 63_6B_31_37_20_00_00_00_00_00_00_00_00_00_00_00_00

Lane 7, 15:

Block 5: D9_C3_33_CC_A0_D3_DC_FF_FF_FF_FF_FF_FF_FF_FF
 Block 6: 5D_2D_CD_0E_C3_23_E1_F0_8F_1D_32_8E_B3_31_39_C2

5. One EIEOSQ (unscrambled) – this resets the scrambler
6. 64 Blocks, each comprising of 16 Symbols of 00h scrambled.
7. One block, unscrambled: 64 UIs of 10b each (voltage level 2 throughout). The scrambler does not advance.
8. One block, unscrambled: 64 UIs of 01b each (voltage level 1 throughout). The scrambler does not advance.
9. 64 Blocks, each comprising of 16 Symbols of 00h scrambled.

The payload in each scrambled Block is the output of the scrambler in that Lane (i.e., input data is 0b). The Lane numbers used to determine the scrambling LFSR seed value depend on how Polling.Compliance is entered. (See § Section 4.2.2.4 for more on assigning Lane numbers and other Scrambler requirements.) The scrambled Blocks of the compliance pattern do not form a Data Stream and hence are exempt from the requirement of transmitting an SDS Ordered Set or the FEC / CRC / Flit formation requirements.

4.2.15 Modified Compliance Pattern in 1b/1b Encoding §

The modified compliance pattern, when not operating in SRIS, consists of repeating the following sequence of 65792 Blocks:

1. One EIEOSQ (that resets the scrambler)
2. 256 Blocks, each comprising of 16 Symbols of 00h scrambled
3. 255 sets of the following sequence:
 - a. One SKP Ordered Set
 - b. 256 Data Blocks, each comprising of 16 Symbols of 00h scrambled

The modified compliance pattern, when operating in SRIS, consists of repeating the following sequence of 67585 Blocks:

1. One EIEOSQ (that resets the scrambler)
2. 2048 sets of the following sequence:
 - a. One SKP Ordered Set
 - b. 32 Blocks, each comprising of 16 Symbols of 00h scrambled

The payload in each scrambled Block is the output of the scrambler in that Lane (i.e., input data is 0b), which are then gray-coded, and precoded, if performed, as shown in § Figure 4-22 . The Lane numbers used to determine the scrambling LFSR seed value depend on how Polling.Compliance is entered. (See § Section 4.2.2.4 for more on assigning Lane numbers.) The scrambled Blocks of the modified compliance pattern do not form a Data Stream and hence are exempt from the requirement of transmitting an SDS Ordered Set or the FEC / CRC / Flit formation requirements.

4.2.16 Jitter Measurement Pattern in 1b/1b Encoding §

The jitter measurement pattern consists of repeating the following 52 UI, consisting of the following 4 sets of 13 UI each:

```
{ 00b, 01b, 00b, 11b, 00b, 10b, 01b, 10b, 11b, 01b, 11b, 10b, 10b,
  00b, 01b, 00b, 11b, 00b, 10b, 01b, 10b, 11b, 01b, 11b, 10b, 00b,
  00b, 01b, 00b, 11b, 00b, 10b, 01b, 10b, 11b, 01b, 11b, 10b, 10b,
  00b, 01b, 00b, 11b, 00b, 10b, 01b, 10b, 11b, 01b, 11b, 10b, 01b }
```

IMPLEMENTATION NOTE: JITTER PATTERN RATIONALE §

The base pattern used above for voltage levels is: { 0, 1, 0, 3, 0, 2, 1, 2, 3, 1, 3, 2 }. This pattern, when repeated, covers all 12 voltage level transitions while being fully DC balanced and using the minimum number of UI.

However, for implementations that use interleaved bit streams, streams, this 12 UI base pattern may not test all the circuits during all the transitions. To address this, a 13th UI is introduced to ensure coverage even with interleaved bitstreams (13 is a prime number). This insertion is done by keeping the same set of transitions and cycling across the 4 sets of values over 52 UI, as shown above. Thus:

- Voltage level 2 (**10b**) is repeated in the 12th and 13th UI of the first row;
- voltage level 0 (**00b**) is repeated in the 13th UI of the second row and the 1st UI of the third row;
- voltage level 3 (**11b**) is repeated in the 11th and 12th UI of the third row; and
- voltage level 1 (**01b**) is repeated in the 10th and 11th UI of the last row.

Across these 52 UI, DC balance is maintained and the DC imbalance between the rows has been minimized by the choice of the sequence of the repeated voltage level.

4.2.17 Toggle Patterns in 1b/1b encoding §

Two types of Toggle Patterns are defined for the best single edge jitter measurement accuracy:

High Swing Toggle Pattern

This comprises of alternating UIs between 00b and 11b (i.e., back to back symbols of 33h unscrambled, effectively alternating between voltage levels 0 and 3 in successive UIs).

Low Swing Toggle Pattern

This comprises of alternating UIs between 01b and 10b (i.e., back to back symbols of 66h unscrambled, effectively alternating between voltage levels 1 and 2 in successive UIs).

4.2.18 Lane Margining at Receiver §

Lane Margining at Receiver, as defined in this section, is mandatory for all Ports supporting a data rate of 16.0 GT/s or higher, including Pseudo Ports (Retimers). Lane Margining at Receiver enables system software to obtain the margin information of a given Receiver while the Link is in the L0 state. The margin information includes both voltage and time, in either direction from the current Receiver position. For all Ports that implement Lane Margining at Receiver, Lane Margining at Receiver for timing is required, while support of Lane Margining at Receiver for voltage is optional at 16.0 GT/s and required at 32.0 GT/s and higher data rates.

Lane Margining at Receiver begins when a Margin Command is received, the Link is operating at 16.0 GT/s Data Rate or higher, and the Link is in L0 state. Lane Margining at Receiver ends when either a Go to Normal Settings command is received, the Link changes speed, or the Link exits either the L0 or Recovery states. Lane Margining at Receiver optionally ends when certain error thresholds are exceeded. Lane Margining at Receiver is permitted to be suspended while the Link is in Recovery for independent samplers.

Lane Margining at Receiver is not supported by Links operating at 2.5 GT/s, 5.0 GT/s, or 8.0 GT/s.

Software uses the per-Lane Margining Lane Control Register and Margining Lane Status Register in each Port (Downstream or Upstream) for sending Margin Commands and obtaining margin status information for the corresponding Receiver associated with the Port. For the Retimers, the commands to get information about the Receiver's capabilities and status and the commands to margin the Receiver are conveyed in the Control SKP Ordered Sets in the Downstream direction. The status and error reporting of the target Retimer Receiver is conveyed in the Control SKP Ordered Sets in the Upstream direction. Software controls margining in the Receiver of a Retimer by writing to the appropriate bits in the Margining Lane Control Register in the Downstream Port. The Downstream Port also updates the status information conveyed by the Retimer(s) in the Link through the Control SKP Ordered Set into its Margining Lane Status Register.

4.2.18.1 Receiver Number, Margin Type, Usage Model, and Margin Payload Fields §

The contents of the four command fields of the Margining Lane Control Register in the Downstream Port are always reflected in the identical fields in the Downstream Control SKP Ordered Sets. The contents of the Upstream Control SKP Ordered Set received in the Downstream Port is always reflected in the corresponding status fields of the Margining Lane Status Register in the Downstream Port. The following table provides the bit placement of these fields in the Control SKP Ordered Set.

Table 4-74 Margin Command Related Fields in the Control SKP Ordered Set §

| Symbol | Description | |
|---|---|--|
| | Usage Model = 0b | Usage Model = 1b |
| $\begin{matrix} \uparrow\downarrow 4^*N \\ +1 \uparrow \end{matrix}$ ECN: Base 6.3 Optical $\Delta \leftrightarrow$ | $\begin{matrix} \uparrow\downarrow \text{Bit 7: Data Parity Bit 6: First Retimer Data Parity Bit 5:} \\ \text{Second Retimer Data Parity Bits [4:0]: Margin CRC} \uparrow \end{matrix}$ | $\begin{matrix} \uparrow\downarrow \text{Bits [7:5]: Reserved Bits} \\ [4:0]: \text{Sideband CRC} \uparrow \end{matrix}$ |
| $4^*N + 2$ | Bit 7: Margin Parity $\begin{matrix} \uparrow\downarrow (\text{see}) \downarrow \end{matrix}$ Bit 6: Usage Model = 0b: Lane Margining at Receiver Bits [5:3]: Margin Type Bits [2:0]: Receiver Number | $\begin{matrix} \uparrow\downarrow \text{Bit 7: Sideband Parity} \uparrow \\ \uparrow\downarrow \text{Bit 6: Usage Model = 1b:} \\ \text{Sideband communication} \uparrow \end{matrix}$ $\begin{matrix} \uparrow\downarrow \text{Bit 5: Sideband Type (0b -} \\ \text{Cmd, 1b - Ack)} \uparrow \end{matrix}$ $\uparrow\downarrow \text{Bits [4:0]:} \uparrow \text{ Reserved}$ |

| Symbol | Description | |
|---------|-----------------------------------|---|
| | Usage Model = 0b | Usage Model = 1b |
| 4*N + 3 | Bits [7:0]: Margin Payload | ↑↓Reserved↓ ↑Bits [7:0]: Sideband Payload ↑ |

↑↑See § Section 4.2.8.2 for definition of Margin CRC / Sideband CRC and Margin Parity / Sideband Parity . CRC and Parity calculations for Usage Model = 1b follow the same bit assignments as Margin CRC and Margin Parity . Having both Usage Model 0 and 1 use the same CRC and Parity assignments allows receivers to still check the integrity of the command, regardless of the Usage Model that they support.↑

ECN: Base 6.3
Optical△↔

Usage Model : An encoding of 0b indicates that the usage model is Lane Margining at Receiver. An encoding of 1b ↑↓in this field↑ indicates that the usage model↑ is ↑↓reserved↓ ↑Sideband communication↑ for ↑↓future usages.↑ ↑PCIe Optical Retimer Solutions (see § Section 4.3).↑

ECN: Base 6.3
Optical△↔

↑↓if↓

IMPLEMENTATION NOTE: POPULATION OF CONTROL SKP ORDERED SET SIDEBAND FIELDS WITH USAGE MODEL SET TO 1 §

↑↑The Control SKP Ordered Set's Usage Model bit for a Lane may be determined by↑ the ↑Lane's↑ Usage Model ↑↓field is 1b, Bits [5:0]↑ ↑bit value↑ of ↑↓Symbol 4N+2↓ ↑its associated Lane N: Margining Control Register Entry . If an implementation chooses to set the Control SKP Ordered Set Usage Model bit using alternative methods, it must ensure that there are no conflicts with the Lane Margining feature. Write↑ and ↑↓Bits [7:0]↑ ↑read-back↑ of ↑↓Symbol 4N+3 are Reserved.↓ ↑the other Control SKP Ordered Set fields associated with sideband communication (Sideband Payload, Sideband Type, and Sideband CRC) is implementation specific.↑

When evaluating received Control SKP Ordered Set for Margin Commands, all Receivers that do not comprehend the usage associated with Usage Model = 1b are required to ignore Bits[5:0] of Symbol 4N+2 and Bits[7:0] of Symbol 4N+3 of the Control SKP Ordered Set , if the Usage Model field is 1b.

IMPLEMENTATION NOTE: POTENTIAL FUTURE USAGE OF CONTROL SKP ORDERED SET §

The intended usage for the 15 bits of information in the Control SKP Ordered Set , as defined in § Table 4-74 is Lane Margining at ↑↓Receiver. However a single bit (Bit 6 of Symbol 4N+2) is Reserved↓ ↑Receiver or Sideband communication↑ for ↑↓any future usage beyond Lane Margining at Receiver. If such a usage is defined in the future, this bit will be set to 1b and the remaining 14 bits can be defined as needed by the new usage model. Alternatively,↓ ↑PCIe Optical Retimer Solution. However,↑ Symbol 4N could use a different encoding than 78h for any future usage, permitting all bits in Symbols 4N+1, 4N+2, and 4N+3 to be defined for that usage model.

Receiver Number : Receivers are identified in § Figure 4-79 . The following Receiver Number encodings are used in the Downstream Port for Margin Commands targeting that Downstream Port or a Retimer below that Downstream Port:

000b

Broadcast (Downstream Port Receiver and all Retimer Pseudo Port Receivers)

001b

Rx(A) (Downstream Port Receiver)

010b

Rx(B) (Retimer X or Z Upstream Pseudo Port Receiver)

011b

Rx(C) (Retimer X or Z Downstream Pseudo Port Receiver)

100b

Rx(D) (Retimer Y Upstream Pseudo Port Receiver)

101b

Rx(E) (Retimer Y Downstream Pseudo Port Receiver)

110b

Reserved

111b

Reserved

The following Receiver Number encodings are used in the Upstream Port for Margin Commands targeting that Upstream Port:

000b Broadcast (Upstream Port Receiver)

001b Reserved

010b Reserved

011b Reserved

100b Reserved

101b Reserved

110b Rx (F) (Upstream Port Receiver)

111b Reserved

Base 6.4 vs Base 6.3

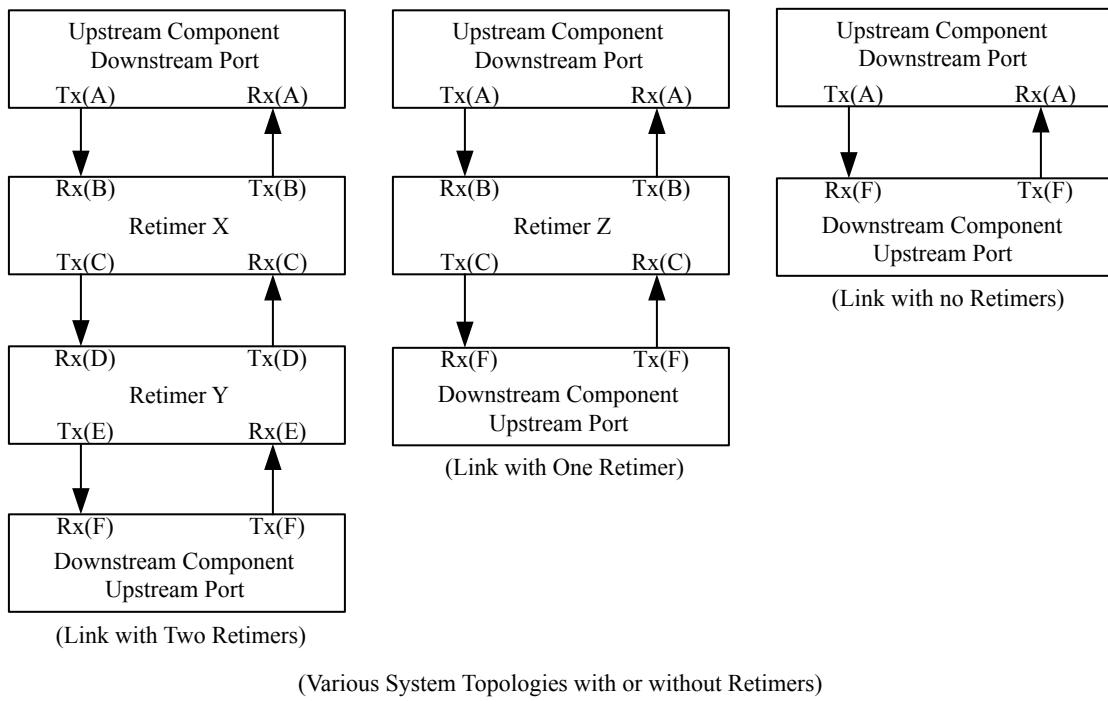


Figure 4-79 Receiver Number Assignment §

Margin Type and Margin Payload : The Margin Type field together with a valid Receiver Number (s), associated with the Margin Type encoding, and specific Margin Payload field define various commands used for margining (referred to as **Margin Command**). § Table 4-75 defines the encodings of valid Margin Commands along with the corresponding responses, used in both the Control SKP Ordered Sets as well as the Margining Lane Control Register and Margining Lane Status Register . Margin commands that are always broadcast will use the broadcast encoding for the Receiver Number , even when only one Receiver is the target (e.g., USP or a DSP in a Link with no Retimers). The Receiver Number field in the response to a Margin Command other than No Command reflects the number of the Receiver that is responding, even for a Margin Command that is broadcast. The Margin Commands go Downstream whereas the responses go Upstream in the Control SKP Ordered Sets . The responses reflect the Margin Type to which the target Receiver is responding. The Receiver Number field of the response corresponds to the target Receiver that is responding. All unused encodings in § Table 4-75 are Reserved and must not be considered to be a valid Margin Command .

The parameters used here (e.g., $M_{sampleCount}$) are defined in § Section 8.4.4 . Each data rate has an independent set of parameters and the values in § Table 4-75 reflect the current data rate. Any relationship between values for different data rates is implementation specific. For example, the timing/voltage step sizes might differ between 64.0 GT/s in PAM-4 mode and 16.0 GT/s or 32.0 GT/s in NRZ mode; N voltage steps at 64.0 GT/s is likely to be a different eye height from N voltage steps at 16.0 GT/s or 32.0 GT/s.

In PAM-4 mode, the Step Margin commands apply to all 3 eyes simultaneously.

Table 4-75 Margin Commands and Corresponding Responses §

| Command | | | | | Response | |
|--|-------------------|---------------------------------|---|-------------------|---|--|
| Margin Command | Margin Type [2:0] | Valid Receiver Number (s) [2:0] | Margin Payload [7:0] | Margin Type [2:0] | Margin Payload [7:0] | |
| No Command | 111b | 000b | 9Ch (No Command is also an independent command in Upstream direction. The expected Response is No Command with the Receiver Number = 000b.) | | | |
| Access Retimer Register | 001b | 010b, 100b | Register offset in bytes: 00h - 87h, A0h - FFh | 001b | Register value, if supported. It is recommended that the Target Receiver on Retimer return 00h if it does not support accessing its registers. It is permitted that the Retimer not respond. | |
| Report Margin Control Capabilities | 001b | 001b through 110b | 88h | 001b | Margin Payload [7:5] = Reserved; Margin Payload [4:0] = { $M_{IndErrorSampler}$, $M_{SampleReportingMethod}$, $M_{IndLeftRightTiming}$, $M_{IndUpDownVoltage}$, $M_{VoltageSupported}$ } | |
| Report $M_{NumVoltageSteps}$ | 001b | 001b through 110b | 89h | 001b | Margin Payload [7] = Reserved Margin Payload [6:0] = $M_{NumVoltageSteps}$ | |
| Report $M_{NumTimingSteps}$ | 001b | 001b through 110b | 8Ah | 001b | Margin Payload [7:6] = Reserved Margin Payload [5:0] = $M_{NumTimingSteps}$ | |
| Report $M_{MaxTimingOffset}$ | 001b | 001b through 110b | 8Bh | 001b | Margin Payload [7] = Reserved Margin Payload [6:0] = $M_{MaxTimingOffset}$ | |
| Report $M_{MaxVoltageOffset}$ | 001b | 001b through 110b | 8Ch | 001b | Margin Payload [7] = Reserved Margin Payload [6:0] = $M_{MaxVoltageOffset}$ | |
| Report $M_{SamplingRateVoltage}$ | 001b | 001b through 110b | 8Dh | 001b | Margin Payload [7:6] = Reserved Margin Payload [5:0] = { $M_{SamplingRateVoltage}$ [5:0] } | |
| Report $M_{SamplingRateTiming}$ | 001b | 001b through 110b | 8Eh | 001b | Margin Payload [7:6] = Reserved Margin Payload [5:0] = { $M_{SamplingRateTiming}$ [5:0] } | |

Base 6.4 vs Base 6.3

| Command | | | | | Response |
|--|-------------------|---------------------------------|--|-------------------|--|
| Margin Command | Margin Type [2:0] | Valid Receiver Number (s) [2:0] | Margin Payload [7:0] | Margin Type [2:0] | Margin Payload [7:0] |
| Report M_{SampleCount} | 001b | 001b through 110b | 8Fh | 001b | Margin Payload [7] = Reserved Margin Payload [6:0] = M _{SampleCount} |
| Report M_{MaxLanes} | 001b | 001b through 110b | 90h | 001b | Margin Payload [7:5] = Reserved Margin Payload [4:0] = M _{MaxLanes} |
| Report Reserved | 001b | 001b through 110b | 91-9Fh | 001b | Margin Payload [7:0] = Reserved |
| Set Error Count Limit | 010b | 001b through 110b | Margin Payload [7:6] = 11b Margin Payload [5:0] = Error Count Limit | 010b | Margin Payload [7:6] = 11b Margin Payload [5:0] = Error Count Limit registered by the target Receiver |
| Go to Normal Settings | 010b | 000b through 110b | 0Fh | 010b | 0Fh |
| Clear Error Log | 010b | 000b through 110b | 55h | 010b | 55h |
| Step Margin to timing offset to right/left of default | 011b | 001b through 110b | See § Section 4.2.18.1.2 | 011b | Margin Payload [7:6] = Step Margin Execution Status (see § Section 4.2.18.1.1) Margin Payload [5:0] = M _{ErrorCount} |
| Step Margin to voltage offset to up/down of default | 100b | 001b through 110b | See § Section 4.2.18.1.2 | 100b | Margin Payload [7:6] = Step Margin Execution Status (see § Section 4.2.18.1.1) Margin Payload [5:0] = M _{ErrorCount} |
| Vendor Defined | 101b | 001b through 110b | Vendor Defined | 101b | Vendor Defined |

Note:

- The term **Step Margin** command is used to refer to either a Step Margin to timing offset to right/left of default or a Step Margin to voltage offset to up/down of default command.

4.2.18.1.1 Step Margin Execution Status §

The **Step Margin Execution Status** used in § Table 4-75 is a 2-bit field defined as follows:

11b

NAK. Indicates that an unsupported Lane Margining command was issued. For example, timing margin beyond ± 0.2 UI. $M_{ErrorCount}$ is 0.

10b

Margining in progress. The Receiver is executing a Step Margin command. $M_{ErrorCount}$ reflects the number of errors detected as defined in § Section 8.4.4.

01b

Set up for margin in progress. This indicates the Receiver is getting ready but has not yet started executing a Step Margin command. $M_{ErrorCount}$ is 0.

00b

Too many errors - Receiver autonomously went back to its default settings. $M_{ErrorCount}$ reflects the number of errors detected as defined in § Section 8.4.4. Note that $M_{ErrorCount}$ might be greater than Error Count Limit.

4.2.18.1.2 Margin Payload for Step Margin Commands §

For the Step Margin to timing offset to right/left of default command, the Margin Payload field is defined as follows:

- Margin Payload [7]: Reserved.
- If $M_{IndLeftRightTiming}$ for the targeted Receiver is Set:
 - Margin Payload [6] indicates whether the Margin Command is right vs. left. A 0b indicates to move the Receiver to the right of the normal setting whereas a 1b indicates to move the Receiver to the left of the normal setting.
 - Margin Payload [5:0] indicates the number of steps to the left or right of the normal setting.
- If $M_{IndLeftRightTiming}$ for the targeted Receiver is Clear:
 - Margin Payload [6]: Reserved
 - Margin Payload [5:0] indicates the number of steps beyond the normal setting.

For the Step Margin to voltage offset to up/down of default command, the Margin Payload field is defined as follows:

- If $M_{IndUpDownVoltage}$ for the targeted Receiver is Set:
 - Margin Payload [7] indicates whether the Margin Command is up vs. down. A 0b indicates to move the Receiver up from the normal setting whereas a 1b indicates to move the Receiver down from the normal setting.
 - Margin Payload [6:0] indicates the number of steps up or down from the normal setting.
- If $M_{IndUpDownVoltage}$ for the targeted Receiver is Clear:
 - Margin Payload [7]: Reserved
 - Margin Payload [6:0] indicates the number of steps beyond the normal setting.

4.2.18.2 Margin Command and Response Flow §

Each Receiver advertises its capabilities as defined in § Section 8.4.4 . The Receiver being margined must report the number of errors that are consistent with data samples occurring at the indicated location for margining. For simplicity, the Margin Commands and requirements are described in terms of moving the data sampler location though the actual margining method may be implementation specific. For example, the timing margin could be implemented on the actual data sampler or an independent error sampler. Further, the timing margin can be implemented by injecting an appropriate amount of stress/jitter to the data sample location, or by actually moving the data/error sample location. When an independent data/error sampler is used, the errors encountered with the independent data/error sampler must be reported in M_ErrorCount even though the Link may not experience any errors. To margin a Receiver, Software moves the target Receiver to a voltage/timing offset from its default sampling position.

The following rules must be followed:

- When operating with 128b/130b encoding:
 - Every Retimer Upstream Pseudo Port Receiver and the Downstream Port Receiver must compute the Margin CRC and Margin Parity bits and compare against the received Margin CRC and Margin Parity bits. Any mismatch must result in ignoring the contents of Symbols 4N+2 and 4N+3. A Downstream Port Receiver must report Margin CRC and Margin Parity errors in the Lane Error Status Register (see § Section 7.7.3.3).
 - The Upstream Port Receiver is permitted to ignore the Margin CRC bits, Margin Parity bits, and all bits in the Symbols 4N+2 and 4N+3 of the Control SKP Ordered Set . If it checks Margin CRC and Margin Parity , any mismatch must be reported in the Lane Error Status Register .
- When operating with the 1b/1b encoding:
 - Every Retimer Upstream Pseudo Port Receiver and the Downstream Port Receiver must compute the Margin CRC bits and compare against the received Margin CRC bits. The Usage Model , Margin Type , Margin Payload and Receiver Number values of the Margin Phy Payload (see § Figure 4-76) within a mismatched copy must be ignored. A Downstream Port Receiver must report Margin CRC errors in the Lane Error Status Register (see § Section 7.7.3.3) when both copies fail Margin CRC.
 - The Upstream Port Receiver is permitted to ignore the Usage Model , Margin CRC , Margin Type , Margin Payload and Receiver Number values of the Margin Phy Payload (see § Figure 4-76). If it checks Margin CRC and both copies fail, the mismatch must be reported in the Lane Error Status Register .
- The Downstream Port must transmit Control SKP Ordered Sets in each Lane, with the Margin Type , Receiver Number , Usage Model , and Margin Payload fields reflecting the corresponding control fields in the Margining Lane Control Register . Any Control SKP Ordered Set transmitted more than 10 μ s after the Configuration Write Completion must reflect the Margining Lane Control Register values written by that Configuration Write.
 - This requirement applies regardless of the values in the Margining Lane Control Register .
 - This requirement applies regardless of the number of Retimer(s) in the Link.
- For Control SKP Ordered Sets received by the Upstream Pseudo Port, a Retimer Receiver is the target of a valid Margin Command , if all of the following conditions are true:
 - the Margin Type is not No Command
 - the Receiver Number is the number assigned to the Receiver, or Margin Type is either Clear Error Log or Go to Normal Settings and the Receiver Number is 'Broadcast'!
 - the Usage Model field is 0b

Errata: Base 6.3
B859 $\Delta\triangleleft$

Errata: Base 6.3
B859 $\Delta\triangleleft$

- the Margin Type , Receiver Number , and Margin Payload fields are consistent with the definitions in § Table 4-74 and § Table 4-75
- the Margin CRC check and Margin Parity check pass.
- For Upstream and Downstream Ports, a Receiver is the target of a valid Margin Command , if all of the following conditions are true for its Margining Lane Control Register :
 - the Margin Type is not No Command
 - the Receiver Number is the number assigned to the Receiver or Margin Type is either Clear Error Log or Go to Normal Settings and the Receiver Number is 'Broadcast'
 - the Usage Model field is 0b
 - the Margin Type , the Receiver Number , and Margin Payload fields are consistent with the definitions in § Table 4-74 and § Table 4-75
- The Upstream Port must transmit the Control SKP Ordered Set with No Command .
- A target Receiver must apply and respond to the Margin Command within 1 ms of receiving the valid Margin Command if the Link is still in L0 state and operating at 16.0 GT/s or higher Data Rate.
 - A target Receiver in a Retimer must send a response in the Control SKP Ordered Set in the Upstream Direction within 1 ms of receiving the Margin Command .
 - A target Receiver in the Upstream Port must update the Status field of the Lane Margin Command and Status register within 1 ms of receiving the Margin Command .
 - A target Receiver in the Downstream Port must update the Status field of the Lane Margin Command and Status register within 1 ms of receiving the Margin Command if the command is not broadcast or no Retimer(s) are present
- For a valid Margin Type , other than No Command , that is broadcast and received by a Retimer:
 - A Retimer, in position X (see § Figure 4-79), forwards the response unmodified in the Upstream Control SKP Ordered Set , if the command has been applied, else it sends the No Command .
 - The Receiver Number field of the response must be set to an encoding of one of the Retimer's Pseudo Ports.
 - The Retimer must respond only after both Pseudo Ports have completed the Margin Command.
- The Retimer must overwrite Bits [4:0] of Symbol 4N+1, Bits[7, 5:0] of Symbol 4N+2 and Bits [7:0] in Symbol 4N+3 as it forwards the Control SKP Ordered Set in the Upstream direction if it is the target Receiver of a Margin Command and is executing the command.
- On receipt of a Control SKP Ordered Set , the Downstream Port must reflect the Margining Lane Status Register from the corresponding fields in the received Control SKP Ordered Set within 1 μ s, if it passes the Margin CRC and Margin Parity checks and one of the following conditions apply:
 - In the Margining Lane Control Register : Receiver Number is 010b through 101b
 - In the Margining Lane Control Register : Receiver Number is 000b, Margin Command is Clear Error Log , No Command , or Go to Normal Settings , and there are Retimer(s) in the Link
 - Optionally, if the Margining Lane Control Register Usage Model field is 1b
 - Optionally, if the Margining Lane Control Register Receiver Number field is 110b or 111b

The Margining Lane Status Register fields are updated regardless of the $\downarrow\downarrow$ Usage Model $\downarrow\downarrow$ $\uparrow\uparrow$ Usage Model $\uparrow\uparrow$ bit in the received Control SKP Ordered Set .
- A component must advertise the same value for each parameter defined in § Table 8-13 in § Section 8.4.4 across all its Receivers. A component must not change any parameter value except for M_{SampleCount} and M_{ErrorCount} defined in § Table 8-13 in § Section 8.4.4 while LinkUp = 1b.

- A target Receiver that receives a valid Step Margin command must continue to apply that offset until any of the following occur:
 - it receives a valid Go to Normal Settings command
 - it receives a subsequent valid Step Margin command with different Margin Type or Margin Payload field
 - M_{IndErrorSampler} is 0b and M_{ErrorCount} exceeds Error Count Limit
 - Optionally, M_{IndErrorSampler} is 1b and M_{ErrorCount} exceeds Error Count Limit.
- If a Step Margin command terminates because M_{ErrorCount} exceeds Error Count Limit, the target Receiver must automatically return to its default sample position and indicate this in the Margin Payload field (Step Margin Execution Status = 00b). Note: termination for this reason is optional if M_{IndErrorSampler} is 1b.
- If M_{IndErrorSampler} is 0b, an error is detected when:
 - The target Receiver is a Port that enters Recovery or detects a Data Parity mismatch while in L0
 - The target Receiver is a Pseudo Port that enters Forwarding training sets or detects a Data Parity mismatch while forwarding non-training sets.
- If M_{IndErrorSampler} is 1b, an error is detected when:
 - The target Receiver is a Port and a bit error is detected while in L0
 - The target Receiver is a Pseudo Port and a bit error is detected while the Retimer is forwarding non-training sets
- If M_{IndErrorSampler} is 0b and either (1) the target Receiver is a Port that enters Recovery or (2) the target Receiver is a Pseudo Port that enters Forwarding training sets:
 - The target Receiver must go back to the default sample position
 - If the target Receiver is a Port that is still performing margining, it must resume the margin position within 128 µs of entering L0
 - If the target Receiver is a Pseudo Port that is still performing margining, it must resume the margin position within 128 µs of Forwarding non-training sets
- A target Receiver is required to clear its accumulated error count on receiving Clear Error Log command, while it continues to margin (if it is the target Receiver of a Step Margin command still in progress), if it was doing so.
- For a target Receiver of a Set Error Count Limit command, the new value is used for all future Step Margin commands until a new Set Error Count Limit command is received.
- If no Set Error Count Limit is received by a Receiver since entering L0, the default value is 4.
- Behavior is undefined if a Set Error Count Limit command is received while a Step Margin command is in effect.
- Once a target Receiver reports a Step Margin Execution Status of 11b (NAK) or 00b ('Too many errors'), it must continue to report the same status as long as the Step Margin command is in effect.
- A target Receiver must not report a Step Margin Execution Status of 01b ('Set up for margin in progress') for more than 100 ms after it receives a new valid Step Margin command
- A target Receiver that reports a Step Margin Execution Status other than 01b, cannot report 01b subsequently unless it receives a new valid Step Margin command.
- Reserved bits in the Margin Payload field must follow these rules:
 - The Downstream or Upstream Port must transmit 0s for Reserved bits
 - The retimer must forward Reserved bits unmodified
 - All Receivers must ignore Reserved bits
- Reserved encodings of the Margin Type, Receiver Number, or Margin Payload fields must follow these rules:

- The retimer must forward Reserved encodings unmodified
- All Receivers must treat Reserved encodings as if they are not the target of the Margin Command
- A Vendor Defined Margin Command or response, that is not defined by a retimer is ignored and forwarded normally.
- A target Receiver on a Retimer must return 00h on the response payload on Access Retimer Register command, if it does not support register access. If a Retimer supports Access Retimer Register command, the following must be observed:
 - It must return a non-zero value for the DWORD at locations 80h and 84h respectively.
 - It must not place any registers corresponding to Margin Payload locations 88h through 9Fh.

4.2.18.3 Flit Mode 8.0 GT/s Margining Behavior §

Lane Margining is defined for 16.0 GT/s and higher data rates. Lane Margining is not supported at 2.5, 5.0, and 8.0 GT/s. In Non-Flit Mode, Control SKP Ordered Sets only present at 16.0 GT/s and higher data rates so margin commands at slower speeds are not possible. In Flit Mode, Control SKP Ordered Sets are also used at 8.0 GT/s.

The following behavior is recommended in Flit Mode at 8.0 GT/s:

- Upstream and Downstream Ports must send Control SKP Ordered Sets as defined, but the margin command should always be No Command, regardless of the contents of the Margining Lane Control Register.
- Upstream Ports, Downstream Ports, and Retimer Upstream Ports should ignore the margin command in received Control SKP Ordered Sets.
 - The Margining Lane Status Register value is unpredictable.
 - Upstream and Downstream Ports are permitted to compute the Margin CRC and Margin Parity bits and compare against the received Margin CRC and Margin Parity bits. If checked, any mismatch must be reported as Margin CRC and Margin Parity errors in the Lane Error Status Register (see § Section 7.7.3.3).

4.2.18.4 Receiver Margin Testing Requirements §

Software must ensure that the following conditions are met before performing Lane Margining at Receiver:

- The current Link data rate must be 16.0 GT/s or higher.
- The current Link width must include the Lanes that are to be tested.
- The Upstream Port's Function(s) must be programmed to a D-state that prevents the Port from entering the L1 Link state. See § Section 5.2 for more information.
- The ASPM Control field of the Link Control register must be set to 00b (Disabled) in both the Downstream Port and Upstream Port.
- The state of the Hardware Autonomous Speed Disable bit of the Link Control 2 register and the Hardware Autonomous Width Disable bit of the Link Control register must be saved to be restored later in this procedure.
- If writeable, the Hardware Autonomous Speed Disable bit of the Link Control 2 register must be Set in both the Downstream Port and Upstream Port. (If hardwired to 0b, the autonomous speed change mechanism is not implemented and is therefore inherently disabled.)
- If writeable, the Hardware Autonomous Width Disable bit of the Link Control register must be Set in both the Downstream Port and Upstream Port. (If hardwired to 0b, the autonomous width change mechanism is not implemented and is therefore inherently disabled.)

While margining, software must ensure the following:

- All Margin Commands must have the Usage Model field in the Margining Lane Control Register set to 0b. While checking for the status of an outstanding Margin Command, software must check that the Usage Model field of the status part of the Margining Lane Status Register is set to 0b.
- Software must read the capabilities offered by a Receiver and margin it within the constraints of the capabilities it offers. The commands issued and the process followed to determine the margin must be consistent with the definitions provided in § Section 4.2.18 and § Section 8.4.4. For example, if the Port does not support voltage testing, then software must not initiate a voltage test. In addition, if a Port supports testing of 2 Lanes simultaneously, then software must test only 1 or 2 Lanes at the same time and not more than 2 Lanes.
- For Receivers where $M_{IndErrorSampler}$ is 1b, any combination of such Receivers are permitted to be margined in parallel.
- For Receivers where $M_{IndErrorSampler}$ is 0b, at most one such Receiver is permitted to be margined at a time. However, margining may be performed on multiple Lanes simultaneously, as long as it is within the maximum number of Lanes the device supports.
- Software must ensure that the Margin Command it provides in the Margining Lane Control Register is a valid one, as defined in § Section 4.2.18.1. For example, the Margin Type must have a defined encoding and the Receiver Number and Margin Payload consistent with it.
- After issuing a command by writing to the Margining Lane Control Register atomically, software must check for the completion of this command. This is done by atomically reading the Margining Lane Status Register and checking that the status fields match the expected response for the issued command (see § Table 4-74 and § Table 4-75). It is strongly recommended that software continue to reread the Margining Lane Status Register if it does not find the expected Receiver Number. If 10 ms has elapsed after a new Margin Command was issued and the values read do not match the expected response, software is permitted to assume that the Receiver will not respond, and declare that the target Receiver failed margining. For a broadcast command other than No Command the Receiver Number in the response must correspond to one of the Pseudo Ports in Retimer Y or Retimer Z, if Retimers are present, or the Downstream Port if they are not, as described in § Figure 4-79.
- Any two reads of the Margining Lane Status Register should be spaced at least 10 μ s apart to make sure they are reading results from different Control SKP Ordered Sets.
- Software must broadcast No Command and wait for it to complete, or for 10 ms to elapse without observing that completion, prior to issuing a new Margin Type or Receiver Number or Margin Payload in the Margining Lane Control Register.
- At the end of margining in a given direction (voltage/ timing and up/down/left/right), software must broadcast Go to Normal Settings, No Command, Clear Error Log, and No Command in series in the Downstream and Upstream Ports, after ensuring each command has been acknowledged by the target Receiver.
- If the Data Rate has changed during margining, margining results (if any) are not accurate and software must exit the margining procedure. Software must set the Margining Lane Control Register to No Command to avoid starting margining if the Data Rate later changes to 16.0 GT/s or higher.
- Software is permitted to issue a Clear Error Log command periodically while margining is in progress, to gather error information over a long period of time.
- Software must not attempt to margin both timing and voltage of a target Receiver simultaneously. Results are undefined if a Receiver receives commands that would place both voltage and timing margin locations away from the default sample position at the same time.
- Software should allow margining to run for at least 10^8 bits margined by the Receiver under test before switching to the next margin step location (unless the error limit is exceeded).

- Software must account for the 'set up for margin in progress' status while measuring the margin time or the number of bits sampled by the Receiver.
- If a target Receiver is reporting 'set up for margin in progress' for 200 ms after issuing one of the Step Margin commands, Software is permitted to assume that the Receiver will not respond and declare that the target Receiver failed margining.
- If a Receiver reports a 'NAK' in the Margin Payload status field and the corresponding Step Margin command was valid and within the allowable range (as defined in § Section 4.2.18 and § Section 8.4.4), Software is permitted to declare that the target Receiver failed margining.
- When the margin testing procedure is completed, the state of the Hardware Autonomous Speed Disable bit and the Hardware Autonomous Width Disable bit must be restored to the previously saved values.

IMPLEMENTATION NOTE: EXAMPLE SOFTWARE FLOW FOR LANE MARGINING AT RECEIVER §

For getting the invariant parameters the following steps may be followed. Once obtained, the same parameters can be used across multiple sets of margining tests as long as LinkUp =1b continues to be true. For each component in the Link, do the following Steps. Software can do these steps in parallel for different components on different Lanes of the Link.

Step A1 :

Issue Report Margin Control Capabilities (Margin Type = 001b, Margin Payload = 88h, Receiver Number = target device in the Margining Lane Control Register)

Step A2 :

Read the Margining Lane Status Register .

- If Margin Type = 001b and Receiver Number = target Receiver: Go to Step A3
- Else: If 10 ms has expired since command issued, declare Receiver failed margining and exit; else wait for >10 µs and Go to Step A2

Step A3 :

Store the information provided Margin Payload status field for use during margining.

Step A4 :

Broadcast No Command (Margin Type = 111b, Receiver Number = 000b, and Margin Payload = 9Ch in the Margining Lane Control Register) and wait for those to be reflected back in the Margining Lane Status Register . If 10 ms expires without getting the command completion handshake, declare the Receiver failed margining and exit.

Step A5 :

Repeat Step A1 through Step A4 for Report M_{NumVoltageSteps} , Report M_{NumTimingSteps} , Report M_{MaxTimingOffset} , Report M_{MaxVoltageOffset} , Report M_{SamplingRateVoltage} , and Report M_{SamplingRateTiming} . It may be noted that this step can be executed in parallel across different Lanes for different Margin Type .

Margining on each Lane across the Link can be a sequence of separate commands. Prior to launching the sequence, software should read the maximum number of Lanes it is allowed to run margining simultaneously. The steps would be similar to Step A1 through Step A4 above with the Report M_{MaxLanes} command. After that software can simultaneously margin up to that many Lanes of the Link. On each Link, each Receiver is margined based on its capability, subject to the constraints described here, after ensuring the Link is operating at full width in 16.0 GT/s or higher Data Rate and the hardware autonomous width and speed change as well as ASPM power states have been disabled.

If software desires to set an Error Count Limit value different than default of 4 or whatever was programmed last, it executes the following Steps prior to going to Step C1 below.

Step B1 :

Issue Set Error Count Limit (Margin Type = 010b, the target Receiver Number , and Margin Payload = {11b, Error Count Limit } in the Margining Lane Control Register)

Step B2 :

Read the Margining Lane Status Register .

- If Margin Type = 010b, Receiver Number = target Receiver, and Margin Payload = Margin Payload control field (Bits [14:7]), go to Step B4
- Else: If 10 ms has expired since command issued, go to Step B3 ; else wait for >10 µs and Go to Step B2

Step B3 :

Margining has failed. Invoke the system checks to find out if the Link degraded in width/speed due to reliability reasons.

Step B4 :

Broadcast No Command and wait for those to be reflected back in the status fields. If 10 ms expires without getting the command completion handshake, declare the Receiver failed margining and exit.

The following steps is an example flow of one margin point for a given Receiver executing Step Margin to timing offset to right/left of default starting with 15 steps to the right:

Step C1 :

Write Margin Type = 011b, the target Receiver Number , and Margin Payload = {0000b, 1111b} in the Margining Lane Control Register

Step C2 :

Read the Margining Lane Status Register .

- If Margin Type = 011b and Receiver Number = target Receiver, Go to Step C3
- Else If 10 ms has expired since command issued, declare Receiver has failed margining and go to Step C7
- Wait for >10 µs and Go to Step C2

Step C3 :

In the Margining Lane Status Register :

- If Margin Payload [7:6] = 11b:
 - If we exceeded the 0.2 UI, that is the margin;
 - Else report margin failure at this point and go to Step C7 ;
- Else if Margin Payload [7:6] = 00b:
 - report margin failure at this point and go to Step C7
- Else if Margin Payload [7:6] = 01b:
 - If 200 ms has elapsed since entering Step C3 , report that the Receiver failed margining test and exit;
 - else wait 1 ms, read the Margining Lane Status Register and go to Step C3
- Else go to Step C4

Step C4 :

Wait for the desired amount of time for margining to happen while sampling the Margining Lane Status Register periodically for the number of errors reported in the Margin Payload field (Bits [5:0] - M_ErrorCount).

For longer runs, issue the No Command followed by the Clear Error Log commands, (using procedures similar to Step B1 through Step B4 , with the corresponding expected status field) if the length of time will cause the error count to exceed the Set Error Count Limit even when staying within the expected BER target.

If the aggregate error count remains within the expected error count and the Margin Payload [7:6] in the status field remains 10b till the end, the Receiver has the required Margin at the timing margin step; else it fails that timing margin step go to Step C7.

Step C5 :

Broadcast No Command and wait for those to be reflected back in the status fields. If 10 ms expires without getting the command completion handshake, declare the Receiver failed margining and exit.

Step C6 :

Go to Step C1 , incrementing the number of timing steps through the Margin Payload control field (Bits[5:0]) if we want to test against a higher margin amount; else go to Step C8 noting the margin value that the Receiver passed

Step C7 :

Margin failed; The previous margin step the Receiver passed in Step C6 is the margin of the Receiver

Step C8 :

Broadcast No Command , Clear Error Log , No Command , Go to Normal Settings series of commands (using a procedure similar to Step B1 through Step B4 with the corresponding expected status fields)

4.3 Retimers §

This Section defines the requirements for Retimers that are Physical Layer protocol aware and that interoperate with any pair of components with any compliant channel on each side of the Retimer. An important capability of a Physical Layer protocol aware Retimer is to execute the Phase 2/3 of the equalization procedure in each direction. A maximum of two Retimers are permitted between an Upstream and a Downstream Port.

The two Retimer limit is based on multiple considerations, most notably limits on modifying SKP Ordered Sets and limits on the time spent in Phase 2/3 of the equalization procedure. To ensure interoperability, platform designers must ensure that the two Retimer limit is honored for all PCI Express Links, including those involving form factors as well as those involving active cables. Form factor specifications may define additional Retimer rules that must be honored for their form factors. Assessing interoperability with any Extension Device not based on the Retimer definition in this section is outside the scope of this specification.

Many architectures of Extension Devices are possible, i.e., analog only Repeater, protocol unaware Retimer, etc. This specification describes a Physical Layer protocol aware Retimer. It may be possible to use other types of Extension Devices in closed systems if proper analysis is done for the specific channel, Extension Device, and end-device pair - but a specific method for carrying out this analysis is outside the scope of this specification.

Retimers have two Pseudo Ports, one facing Upstream, and the other facing Downstream. The Transmitter of each Pseudo Port must derive its clock from a 100 MHz reference clock. The reference clock(s) must meet the requirements of § Section 8.6 . A Retimer supports one or more reference clocking architectures as defined in § Section 8.6 Electrical Sub-block.

In most operations Retimers simply forward received Ordered Sets, DLLPs, TLPs, Logical Idle, ~~↑↓and↓~~ Electrical ~~↑↓Idle, ↓↑Idle, and Flits.↑~~ Retimers are completely transparent to the Data Link Layer and Transaction Layer. System software shall not enable L0s on any Link where a Retimer is present. Support of Beacon by Retimers is optional and beyond the scope of this specification.

When using 128b/130b encoding the Retimer executes the protocol so that each Link Segment undergoes independent Link equalization as described in § Section 4.3.7 .

The Pseudo Port orientation (Upstream or Downstream) is determined dynamically, while the Link partners are in Configuration. Both crosslink and regular Links are supported.

ECN: Base 6.3
Optical△

↑↑↑Implementations that support Flit Mode and data rates of 64.0 GT/s or higher are permitted to deploy optical technologies as part of a Link extension solution. In this type of deployment, instead of one extension component having two PCIe protocol aware Pseudo Ports, as is the case with purely electrical Retimers, the entire extension solution has two PCIe protocol aware Pseudo Ports with an optical technology-based Link extension mechanism in between the protocol aware Pseudo Ports. This extension implementation is an Optical Retimer Solution. A component that has an electrical Pseudo Port on one end and an optical technology-based port on the other end is an Optical Aware Retimer. The requirements for the protocol aware Pseudo Ports are defined, in an optical technology neutral manner, in this specification. The operation of the extension mechanism in between the two electrical Pseudo Ports (i.e., the optical technology, the conversion of the signals from electrical to optical or optical to electrical, the specific optical modulation, wavelength, power and fiber type used) is implementation specific. § Figure 4-81 illustrates a topology using an Optical Retimer Solution and identifies the portions of the implementation (the Electrical Link Segments) that must be PCIe compliant.↑

4.3.1 Retimer Requirements §

The following is a high level summary of Retimer requirements:

- ↑↑When optical technologies are deployed as part of a Link extension solution as illustrated in § Figure 4-81 :↑
 - ↑↑The Link extension solution is an Optical Retimer Solution.↑
 - ↑↑Retimer rules apply to the Pseudo Ports connected to the Electrical Link Segments only.↑
 - ↑↑The requirements of this section (and other sections that are referenced in this section) apply to the electrical ports that are operating as the Upstream Pseudo Port and the Downstream Pseudo Port only.↑
 - ↑↑Requirements of the optical portion of the Optical Retimer Solution are implementation specific unless otherwise specified.↑
- Retimers are required to comply with all the electrical specification described in § Chapter 8. Electrical Sub-block. Retimers must operate in one of two modes:
 - Retimers' Receivers operate at 8.0 GT/s and above with an impedance that meets the range defined by the Z_{RX-DC} parameter for 2.5 GT/s.
 - Retimers' Receivers operate at 8.0 GT/s and above with an impedance that does not meet the range defined by the Z_{RX-DC} parameter for 2.5 GT/s. In this mode the Z_{RX-DC} parameter for 2.5 GT/s must be met with in 1 ms of receiving an EOS or inferring Electrical Idle and while the Receivers remain in Electrical Idle.
- Forwarded Symbols must always be de-skewed when more than one Lane is forwarding Symbols (including upconfigure cases).
- Determine Port orientation dynamically.
- Determine Data Stream Mode (Flit Mode or Non-Flit Mode) dynamically.
- Perform Lane polarity inversion (if needed).
- Interoperate with the Link equalization procedure for Phase 2 and Phase 3, when using 128b/130b or 1b/1b encoding, on each Link Segment.
- Interoperate with de-emphasis negotiation at 5.0 GT/s, on each Link Segment.
- Interoperate with Link Upconfigure.
- Interoperate with L0p .
- Pass loopback data between the Loopback Lead and Loopback Follower .

- Optionally execute Follower Loopback on one Pseudo Port when using 8b/10b or 128b/130b encoding
- Execute Follower Loopback on one Pseudo Port when using 1b/1b
- Generate the Compliance Pattern on each Pseudo Port.
 - Load board method (i.e., time out in Polling.Active).
- Forward Modified Compliance Pattern when the Link enters Polling.Compliance via Compliance_Receive_Request in TS1 Ordered Sets.
- Forward Compliance or Modified Compliance Patterns when Ports enter Polling.Compliance via the Enter Compliance bit in the Link Control 2 register is set to 1b in both the Upstream Port and the Downstream Port and Retimer Enter Compliance is set to 1b (accessed in an implementation specific manner) in the Retimer.
- Adjust the data rate of operation in concert with the Upstream and Downstream Ports of the Link.
- Adjust the Link width in concert with the Upstream and Downstream Ports of the Link.
- Capture Lane numbers during Configuration .
 - Lane numbers are required when using 128b/130b and 1b/1b encoding for the scrambling seed.
- Capture the Flit Mode Supported bit during Polling and Configuration during Link training.
 - The Flit Mode Supported bit is used to determine the Data Stream Mode.
 - ↑↑The Flit Mode Supported bit is used to determine the definition of the Data Rate Identifier.↑↑
 - ↑↑Optical Aware Retimers (Retimers deploying optical technologies) must support Flit Mode.↑↑
- Dynamically adjust Retimer Receiver impedance to match end Component Receiver impedance.
- Infer entering Electrical Idle at all data rates.
- Modify certain fields of Ordered Sets while forwarding.
- Perform clock compensation via addition or removal of SKP Symbols.
- Support L1 .
 - Optionally Support L1 PM Substates .
- If 32.0 GT/s capable, then interoperate with Link equalization to the highest data rate.
- If 32.0 GT/s capable, then interoperate with No Equalization Needed mode.
- If 32.0 GT/s capable, then interoperate with the use of ↓↓Modified TS1/TS2 Ordered Sets.↓↓ ↑↑Modified TS1/TS2 Ordered Sets .↑↑
- Forward 1b/1b Control SKP Ordered Sets that have Phy Payload Type equal to 0b. Thus, 1b/1b Control SKP Ordered Sets with Margin Payload ↑↑or Sideband communication↑↑ will be forwarded.

ECN: Base 6.3
Optical△↔ECN: Base 6.3
Optical△↔

4.3.2 Supported Retimer Topologies §

§ Figure 4-80 shows the topologies supported by Retimers defined in this specification. There may be one or two Retimers between the Upstream and Downstream Ports on a Link. Each Retimer has two Pseudo Ports, which determine their Downstream/Upstream orientation dynamically. Each Retimer has an Upstream Path and a Downstream Path. Both Pseudo Ports must always operate at the same data ↓↓rate,↓↓ ↑↑rate↑↑ when in Forwarding mode. Thus, each Path will also be at the same data rate. A Retimer is permitted to support any width option defined by this specification as its maximum width.

ECN: Base 6.3
Optical△↔

↑↑Link extension solutions that deploy optical technologies are strongly recommended to support the two Retimer topology illustrated in § Figure 4-81 . That figure illustrates an Optical Retimer Solution, the extension of a Link with two extension components, each of which has an electrical Port connected to an optical technology. The extension components are Optical Aware Retimers. During Link training, the electrical Port of one Optical Aware Retimer will determine that it must operate as an Upstream Pseudo Port and the other electrical Port (of the other Optical Aware Retimer) will determine that it must operate as a Downstream Pseudo Port. The electrical Pseudo Ports must meet the Retimer requirements (specified in this document) for their defined direction (i.e., the designated Upstream Pseudo Port of the Optical Retimer Solution must meet the Retimer Upstream Pseudo Port requirements; and the designated Downstream Pseudo Port must meet the Retimer Downstream Pseudo Port requirements). The Optical Retimer Solution appears as if there is one Retimer extending the Link, with one Upstream Pseudo Port and one Downstream Pseudo Port. An implementation that deploys optical technologies as part of an extension solution but has a different topology than that illustrated in § Figure 4-81 is permitted but not specified. If deployed, such a topology must ensure that the requirements of both PCIe electrical Link segments are met.↑

IMPLEMENTATION NOTE: LINK RETIMER LIMIT §

ECN: Base 6.3
Optical△↔

↑↑An Optical Retimer Solution consists of two Optical Aware Retimers. An implementation that requires additional Retimers must deploy components that are Four Retimer Aware (FRA). More specifically, the other Retimers must be FRA and, ideally, the Downstream and Upstream Components at each end of the Link must also be aware of the extension topologies that deploy more than two Retimers.↑

The behavior of the Retimer in each high level operating mode is:

- ↑↑Activation mode:
 - ↑↑Applies to Optical Aware Retimers.↑
 - ↑↑Consists of turning on the electrical receiver's impedance after the optical link between the Optical Aware Retimers has been established/activated.↑
 - ↑↑Performed in three phases: Detect , Detect.Active2 , and Detect.Active3↑
- Forwarding mode:
 - Symbols, Electrical Idle, and exit from Electrical Idle; are forwarded on each Upstream and Downstream Path.
- Execution mode:
 - The Upstream Pseudo Port acts as an Upstream Port of a Component. The Downstream Pseudo Port acts as a Downstream Port of a Component. This mode is used in the following cases:
 - Polling.Compliance .
 - Phase 2 and Phase 3 of the Link equalization procedure.
 - Optionally Follower Loopback.

Base 6.4 vs Base 6.3

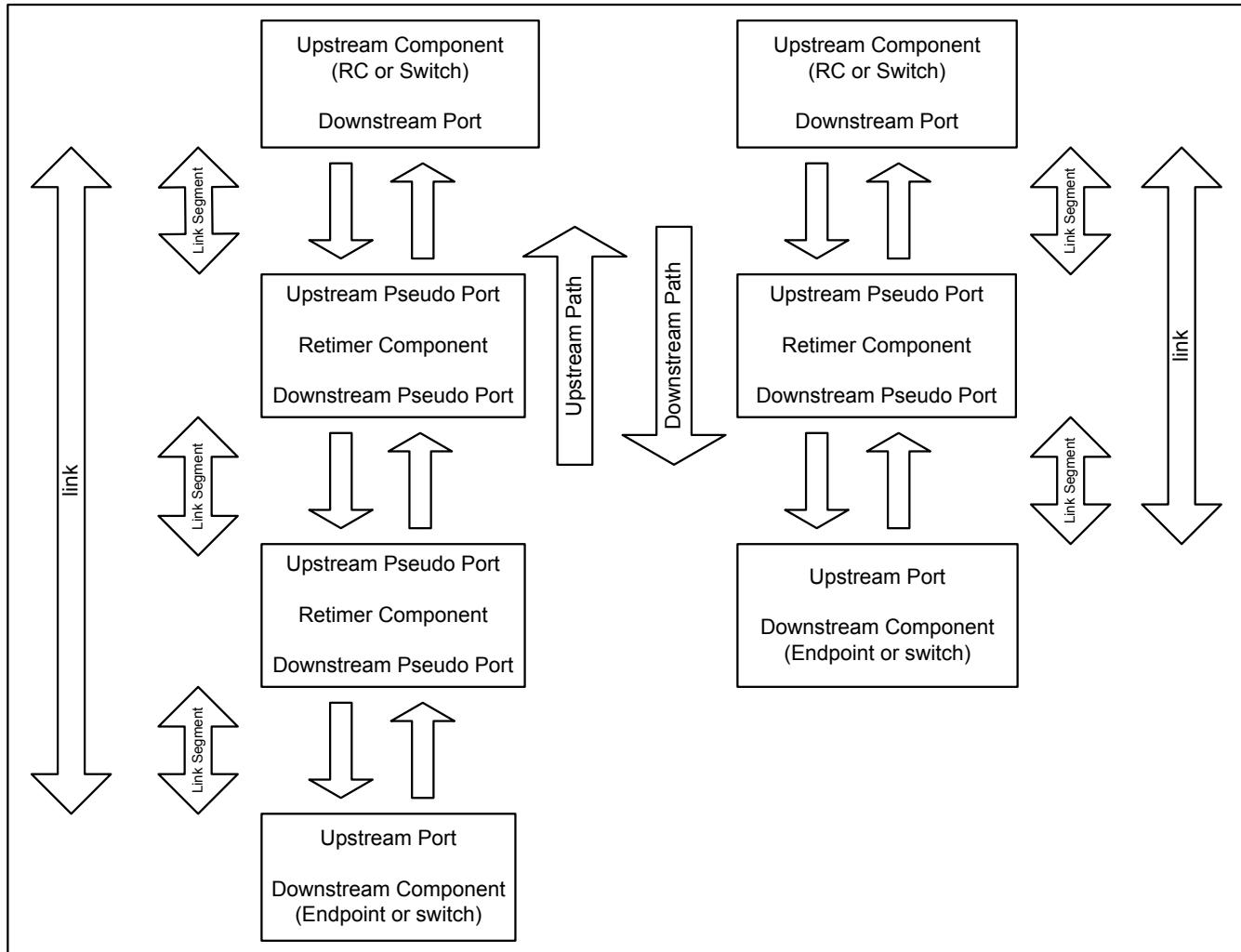


Figure 4-80 Supported Retimer Topologies §

Base 6.4 vs Base 6.3

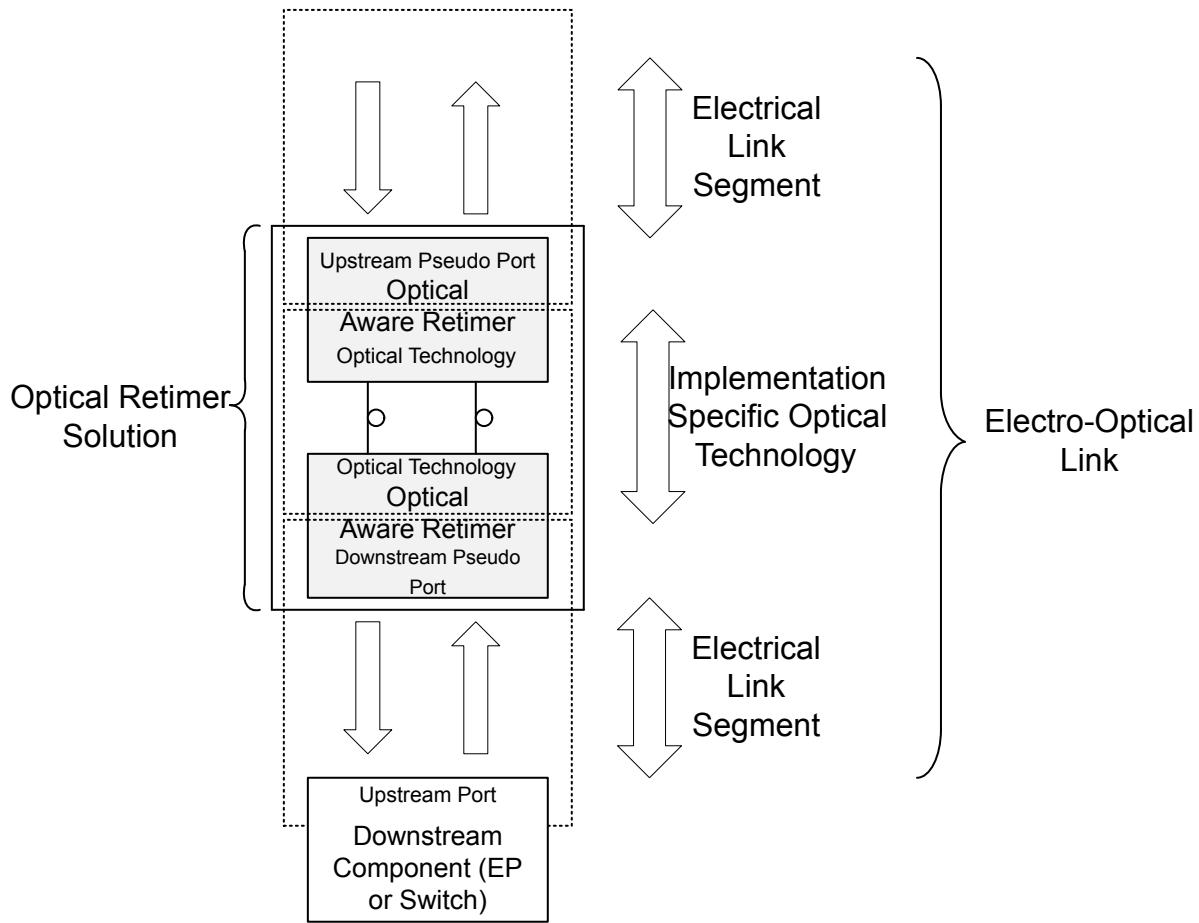


Figure 4-81 [Supported Retimer Topology Using an Optical Retimer Solution](#)

ECN: Base 6.3
Optical

4.3.3 Variables

The following variables are set to the following specified values following a Fundamental Reset or whenever the Retimer receives Link and Lane number equal to PAD on two consecutive TS2 Ordered Sets on all Lanes that are receiving TS2 Ordered Sets on both Upstream and Downstream Pseudo Ports within a 1 µs time window from the last Symbol of the second TS2 Ordered Set on the first Lane to the last Symbol of the second TS2 Ordered Set on the last Lane.

- ***RT_port_orientation*** = undefined
- ***RT_captured_lane_number*** = PAD
- ***RT_captured_link_number*** = PAD
- ***RT_G3_EQ_complete*** = 0b
- ***RT_G4_EQ_complete*** = 0b
- ***RT_G5_EQ_complete*** = 0b
- ***RT_G6_EQ_complete*** = 0b
- ***RT_LinkUp*** = 0b
- ***RT_number*** = undefined
- ***RT_next_data_rate*** = 2.5 GT/s

- ***RT_error_data_rate*** = 2.5 GT/s
- ***RT_flit_mode_enabled*** = 0b

4.3.4 Receiver Impedance Propagation Rules §

The Retimer Transmitters and Receivers shall meet the requirements in § Section 4.2.5.9.1 while Fundamental Reset is asserted. When Fundamental Reset is deasserted the Retimer is permitted to take up to 100 ms to begin active determination of its Receiver impedance. A Retimer that supports only Link speeds 5.0 GT/s or less must do this within 20 ms. During this interval the Receiver impedance remains as required during Fundamental Reset. Once this interval has expired Receiver impedance on Retimer Lanes is determined as follows:

- Within 1.0 ms of the Upstream or Downstream Port's Receiver meeting the ***Z_{RX-DC}*** parameter, the low impedance is back propagated, (i.e., the Retimer's Receiver shall meet the ***Z_{RX-DC}*** parameter on the corresponding Lane on the other Pseudo Port). Each Lane operates independently and this requirement applies at all times.
- The Retimer must keep its Transmitter in Electrical Idle until the ***Z_{RX-DC}*** condition has been detected. This applies on an individual Lane basis.

↑↑When an Optical Aware Retimer is deployed, Receiver impedance on Retimer Lanes takes effect once Activation mode (i.e., Detect.Active, Detect.Active2, and Detect.Active3) is complete.↑

ECN: Base 6.3
Optical△↔

4.3.5 Switching Between Modes §

The Retimer operates in ↑↑two+↑↑three↑ basic modes, ↑↑Activation Mode,↑ Forwarding ↑↓mode↓↑↑mode,↑ or Execution mode. ↑↑Activation mode only applies to Optical Aware Retimers.↑ When switching between ↑↑these↓↑↑the Forwarding and Execution↑ modes the switch must occur on an Ordered Set boundary for all Lanes of the Transmitter at the same time. No other Symbols shall be between the last Ordered Set transmitted in the current mode and the first Symbol transmitted in the new mode.

ECN: Base 6.3
Optical△↔

When using 128b/130b or 1b/1b the Transmitter must maintain the correct scrambling seed and LFSR value when switching between modes.

When switching between Forwarding and Execution modes, the Retimer must ensure that at least 16 TS0 / TS1 Ordered Sets and at most 64 TS0 / TS1 Ordered Sets are transmitted between the last EIEOS transmitted in the previous mode and the first EIEOS transmitted in the new mode.

When switching to and from the Execution Link Equalization mode the Retimer must ensure a Transmitter does not send two SKP Ordered Sets in a row, and that the maximum allowed interval is not exceeded between SKP Ordered Sets, see § Section 4.2.8.4.

4.3.6 ↑↑Activation Rules↑ §

ECN: Base 6.3
Optical△↔

↑↑Activation mode is used to sequence receiver detection when Optical Aware Retimers are being used as part of the Link extension implementation. During this mode, the Retimers' receivers (i.e., the Pseudo Port on the electrical end of the Optical Aware Retimer) will appear offline until the optical portion of the Optical Retimer Solution has been established and is ready for data transfer. The following rules apply after the deassertion of Fundamental Reset:↑

Base 6.4 vs Base 6.3

- ↑↑ Detect.Active phase: Optical Aware Retimers perform receiver detection on their Pseudo Ports connected to the Electrical Link Segments following the deassertion of Fundamental Reset, but they do not turn on their receivers. Transmit direction behavior is equivalent to the Transmitter behavior in Detect.Active .↑
- ↑↑ Once receivers have been detected, the Optical Aware Retimer moves to the Detect.Active2 phase of Activation mode. In this phase, the optical Link is established/activated. This phase requires that both Pseudo Ports (of the Optical Retimer Solution) connected to the Electrical Link Segments of the Link extension implementation have entered Detect.Active2 . The method for determining entry into Detect.Active2 (by the appropriate Pseudo Ports) is implementation specific. Behavior is undefined if either Pseudo Port does not detect any receivers.↑
- ↑↑ Once the optical Link is established/activated, each end communicates the number of receivers that were detected on the Electrical Link Segments to the other end of the Link extension implementation (each Optical Aware Retimer will know the number of Receivers detected by the Optical Aware Retimer at the other end of the Optical Retimer Solution). This information will be used in the Detect.Active3 phase to avoid unnecessarily turning on receivers that will not be used for this Link.↑
- ↑↑ Optical Aware Retimer moves to Detect.Active3 phase and turns on the same number of receivers that were detected by the Pseudo Port connected to the opposite Electrical Link Segment (and communicated to this Pseudo Port in the previous step).↑
- ↑↑ In Detect.Active3 , the Optical Aware Retimer forwards TS1 Ordered Sets with Link and Lane numbers set to PAD (as is done in Polling.Active).↑
- ↑↑ Next mode is Forwarding Mode when Electrical Idle is exited on all detected Lanes or 1 ms expires and 8 consecutive TS1 Ordered Sets are received on any Lane.↑
- ↑↑ During Detect.Active3 phase, Optical Aware Retimers must use implementation specific means to ensure that the optical connection remains active/calibrated↑

IMPLEMENTATION NOTE: EXAMPLE OF AN OPTICAL AWARE RETIMER EXTENSION IMPLEMENTATION §

↑↑An example optical extension implementation will be defined with two Optical Aware Retimers connected as illustrated in § Figure 4-83 (i.e., an Optical Retimer Solution). Other topologies (with other than two Retimers) are not precluded. However, this implementation note illustrates a possible solution with two Optical Aware Retimers (§ Figure 4-83) and the steps taken to establish the optical link extension implementation.↑

1. ↑↑Fundamental Reset is de-asserted. This could happen across the entire system or one component at a time. The sequence of de-asserting Fundamental Reset is implementation specific.↑
2. ↑↑Components enter Detect.Active (after Fundamental Reset and any other conditions required to transition are met).↑
3. ↑↑In Detect.Active , Optical Aware Retimers perform receiver detection on local Electrical Link Segments but do not turn on their receivers for detection.↑
4. ↑↑Once receivers are detected, the Optical Aware Retimer moves to Detect.Active2 ; begins to establish the optical channel.↑
 - ↑↑Establish/activation of the optical channel is performed after Detect.Active2 is entered by both sides of the optical channel↑
 - ↑↑Establish/activation of the optical channel is implementation specific and, in this example, the highest operating rate for the optical portion of the link will be achieved. Some optical technologies may be able to match the PCIe Link speed, in which case the maximum rate is not required in this step.↑
 - ↑↑Subsequent to the establishment/activation of the optical channel and prior to the transition to Detect.Active3 , the number of lanes detected should be communicated across the optical channel to the “other” Optical Aware Retimer. This information will be used in Detect.Active3 .↑
5. ↑↑The Optical Aware Retimer moves to Detect.Active3 . The Retimer’s receivers are enabled for detection. The number of receivers turned on is limited to the number of receivers detected by the “other” Optical Aware Retimer in Detect.Active (information which was communicated at the end of Detect.Active2). This prevents turning on or mapping lanes that will never be used.↑
 - ↑↑The Optical Aware Retimer forwards TS1/PAD/PAD.↑
 - ↑↑The Optical Aware Retimer transitions to Forwarding Mode when El exited on all detected lanes or 1 ms expires and 8 TS1s received on any lane.↑
 - ↑↑The Optical Aware Retimers use implementation specific means to ensure that the optical channel remains active.↑
6. ↑↑Once in Forwarding Mode, the Optical Aware Retimer forwards received TS OS across the optical channel to far end.↑
 - ↑↑The optical side of the Optical Aware Retimer is aware of both the number of electrical lanes and optical channels and performs the lane to channel mapping.↑

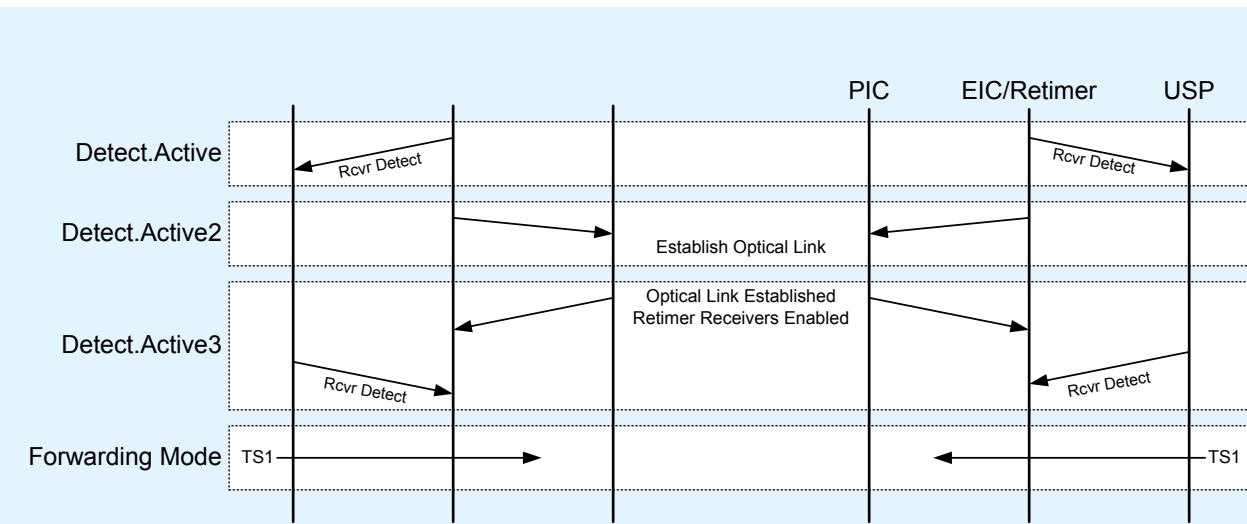


Figure 4-82 Optical Aware Retimer Example – Flow Diagram of Steps 1 to 6

- ↑↑↑The Optical Aware Retimer must store the Flit Mode Supported bit the from TS1s it received from optical channels while the Link is in Polling.Active .↑
- ↑↑↑The Optical Aware Retimer must store the Supported Link Speeds value from the TS2s it received from the optical channels while the Link is in Polling.Configuration.↑
- ↑↑↑Checks for optical during the TS1 / TS2 exchange in Polling and Configuration .↑
 - ↑↑↑See Implementation Note: Sideband Communication with 8b/10b Encoded Rates .↑
- ↑↑↑The Optical Aware Retimer stores the outcomes typically associated with Link training – Link/Lane numbers, APN results, maximum common supported rates, etc.↑
- ↑↑↑Bypass EQ to the highest NRZ rate or No Equalization Needed is negotiated.↑
- ↑↑↑Link enters L0 at 2.5 GT/s.↑
- ↑↑↑Root Complex initiates speed change to highest NRZ rate (when performing EQ).↑
 - ↑↑↑TS1 / TS2 continue to be sent end to end↑
 - ↑↑↑When the electrical end of the Optical Aware Retimer is the upstream pseudo port, Tx Preset/Coefficient in TS2s received in Recovery.RcvrCfg are stored for use once transmission begins at negotiated rate.↑
 - ↑↑↑When the electrical end of the Optical Aware Retimer is the Downstream Pseudo Port, the Retimer may either forward Recovery.RcvrCfg EQ TS2s unmodified or override the values with its own preferred initial settings for the Link partner's Transmitters.↑
- ↑↑↑Speed change happens on the electrical links only (i.e., Root Complex/Retimer 1 and End Point/Retimer 2).↑
 - ↑↑↑Transmitters on the electrical side of the Optical Aware Retimer enter EI when the Receivers on the optical side of the Retimer indicate that it has determined that electrical idle has been entered on the corresponding electrical link (i.e., Tx(B) can enter EI when it “understands” that Tx(F) has entered EI) – the criteria to perform this determination are implementation specific.↑
 - ↑↑↑Any E2O or O2E operational changes resulting from speed changes will be handled by optical side of the Optical Aware Retimers (and are implementation specific).↑

15.

↑↑↑EI exit on the Optical Aware Retimer electrical side transmitters occurs when TS1 OS arrive on optical side receivers.↑

 - ↑↑↑For example, Tx(E) exits after sets from Tx(A) transmitted, which eventually arrive at Rx(D); Tx(B) exits after sets from Tx(F) transmitted, which eventually arrive at Rx(C).↑
 - ↑↑↑Speed changes are performed while observing the time requirements of Recovery .↑
 16.

↑↑↑Equalization is initiated by Root Complex and performed as it typically would be in a fully electrical Retimer.↑

 - ↑↑↑Information about Electrical links must be conveyed via TS1s across optical channel to ensure that the Retimer Equalization Extend (REE) bit is set appropriately.↑
 - ↑↑↑For example, in Phase 2, Rx(B) must set REE to 1b until Rx(F) → Tx(E) equalization is complete and the End Point moves to Phase 3↑
 - ↑↑↑Example 2, in Phase 3, Rx(E) must set REE to 1b until Rx(A) → Tx(B) equalization is complete and Root Complex moves to Recovery.RcvrLock .↑
 17.

↑↑↑Upon completion of equalization, steps 13-16 are repeated if speed change and equalization at a higher rate are required.↑

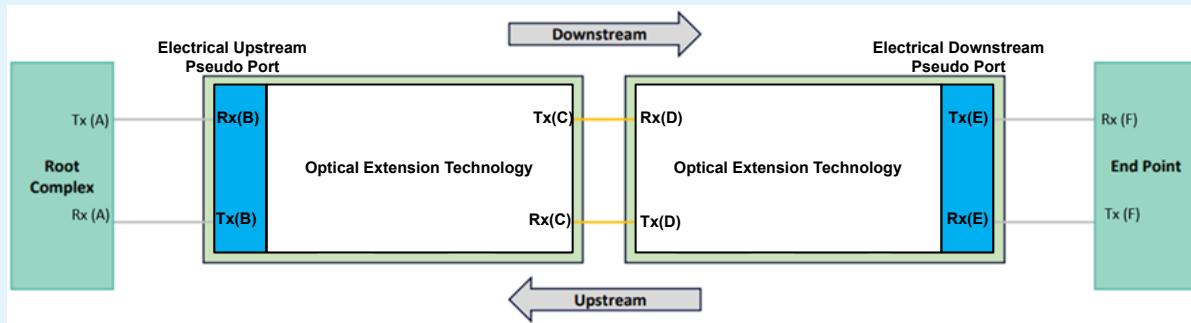


Figure 4-83 ↑Sample Retimer Topology Using PCIe Optical Aware Retimers↑ §

4.3.7 Forwarding Rules §

These rules apply when the Retimer is in Forwarding mode. The Retimer is in Forwarding mode after \uparrow Activation mode when it is an Optical Aware Retimer. Otherwise, after \uparrow the deassertion of Fundamental Reset.

- If the Retimer's Receiver detects an exit from Electrical Idle on a Lane the Retimer must enter Forwarding mode and forward the Symbols on that Lane to the opposite Pseudo Port as described in § Section 4.3.7.3 . ↑↑In Forwarding mode, the detection of exit from Electrical Idle has already been achieved by Optical Aware Retimers because it is a requirement for transition from Activation mode to Forwarding mode.↑
 - The Retimer must continue to forward the received Symbols on a given Lane until it enters Execution mode or until an EIOS is received, or until Electrical Idle is inferred on that Lane. This requirement applies even if the Receiver loses Symbol lock or Block Alignment. See § Section 4.3.7.5 for rules regarding Electrical Idle entry.
 - ↑↑Optical Aware Retimers are required to perform the appropriate electrical Lane to optical channel mappings when the numbers of each quantity do not match. The optical bit error rates must be equal to or better than the electrical bit error rates (i.e., FBER=10↑
↑↑-6↑↑).↑

ECN: Base 6.3
Optical $\Delta \ll$

ECN: Base 6.3
Optical AEP

Execution mode

Base 6.4 vs Base 6.3

- A Retimer shall forward all Symbols unchanged, except as described in § Section 4.3.7.9 and § Section 4.3.7.7 .
- When operating at 64.0 GT/s data rate, a Retimer must follow the requirements of § Section 4.2.3.2 in order to identify SKP OS , EIEOS , and EIOS received.
- When operating at 2.5 GT/s data rate, if any Lane of a Pseudo Port receives TS1 Ordered Sets with Link and Lane numbers set to PAD for 5 ms or longer, and the other Pseudo Port does not detect an exit from Electrical Idle on any Lane in that same window, ↑↓the Retimer enters the Execution mode CompLoadBoard state,↑ and ↑↓follows § Section 4.3.8.1 if↑ either of the following occurs:
 - The following sequence occurs:
 - An EIOS is received on any Lane that was receiving TS1 Ordered Sets
 - followed by a period of Electrical Idle, for less than 5 ms
 - followed by Electrical Idle Exit that cannot be forwarded according to § Section 4.3.7.3
 - Note: this is interpreted as the Port attached to the Receiver going into Electrical Idle followed by a data rate change for a Compliance Pattern above ↑↓2.5 GT/s.↓ ↑↓2.5 GT/s (as described in § Section 4.2.7.2.2).↑
 - Compliance Pattern at 2.5 GT/s is received on any Lane that was receiving TS1 Ordered Sets.

Errata: Base 6.3
B816△◀▷

- ↑↓Then the Retimer enters the Execution mode CompLoadBoard state, and follows↓ ↑↓.↓ ↑↑↑ Errata: Base 6.3
B816△◀▷
- If any Lane on the Upstream Pseudo Port receives two consecutive TS0 / TS1 Ordered Sets with the EC field equal to 10b, when using 128b/130b or 1b/1b encoding, then the Retimer enters Execution mode Equalization, and follows § Section 4.3.8.2 .
 - If the Retimer is configured to support Execution mode Follower Loopback and if any Lane on either Pseudo Port receives two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets with Loopback_Request asserted then the Retimer enters Execution mode Follower Loopback, and follows § Section 4.3.8.3 .

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: MAPPING OF ELECTRICAL LANES TO OPTICAL CHANNELS WHEN USING OPTICAL EXTENSION TECHNOLOGIES §

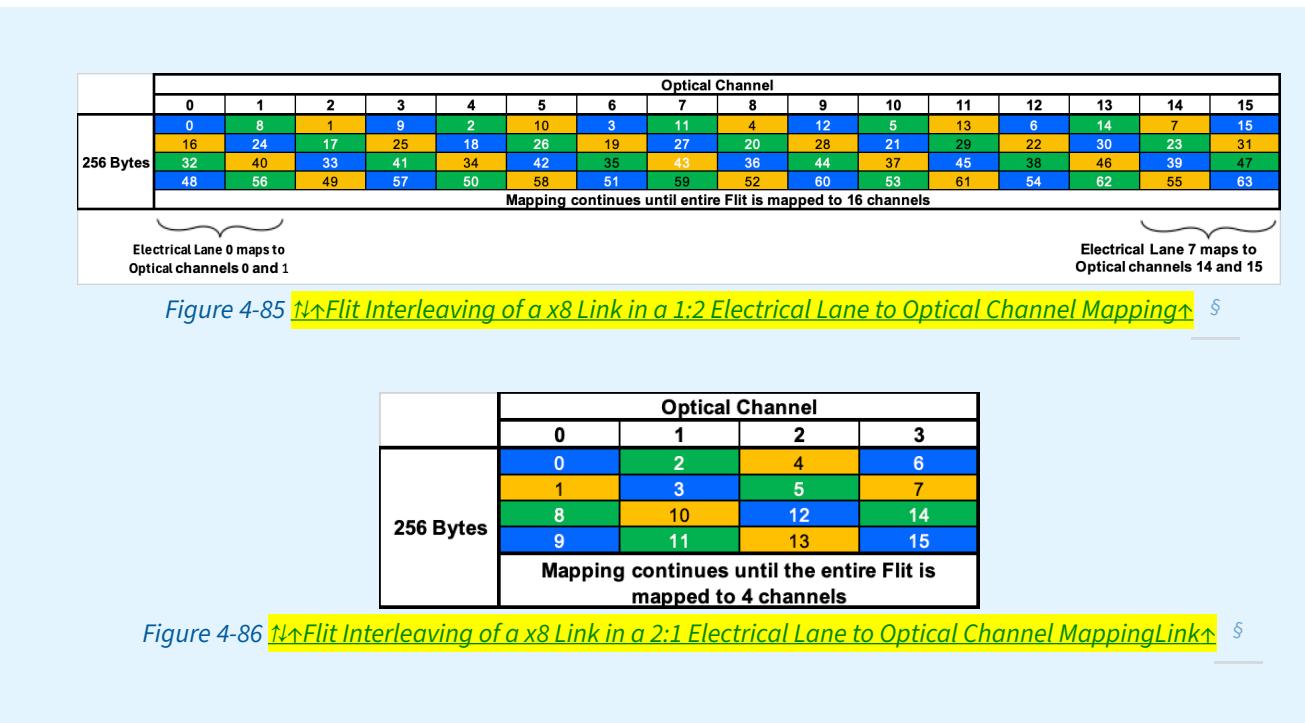
↑↑The number of electrical Lanes may not match the number of optical channels when using an optical extension technology implementation. This may be because of the Electrical Link Segments being trained to less than the maximum width of the extension implementation or by design. When this mismatch occurs, the following guidelines are suggested:↑

ECN: Base 6.3
Optical△↔

- ↑↑One electrical Lane may be mapped to one or more optical channels (1:1, 1:2, 1:4, 1:8).↑
- ↑↑OR multiple electrical Lanes may be mapped to one optical channel (2:1, 4:1, 8:1).↑
- ↑↑Layout is the same for electrical or optical channel.↑
- ↑↑Byte i is sent on Lane/ channel ($i \bmod m$) where m is the number of Lanes or channels.↑
- ↑↑Byte-interleaved – so Retimer needs to do the 1:n or n:1 mapping after de-skew (1:1 does not need de-skew).↑
- ↑↑Ordered Set after LinkUp=1b follow same mapping with byte interleaving.↑

| Description | | Lane | | | | | | | |
|------------------------------------|--------------------|----------------|-------|-------|----------|----------|----------|----------|----------|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 256 Symbol Times (i.e., 256 Bytes) | TLP Bytes [0..231] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| | | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| | | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| | | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| | | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| | | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| | | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| | | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| | | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| | | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| | | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| | | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| | | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| | | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| | | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 |
| | | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| | | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| | | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| | | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| | | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| | | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| | | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| | | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 |
| | | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| | | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 |
| | | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| | | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 |
| | | 232 | 233 | 234 | 235 | DLP 0 | DLP 1 | DLP 2 | DLP 3 |
| | | DLP, CRC Bytes | DLP 4 | DLP 5 | CRC 0 | CRC 1 | CRC 2 | CRC 3 | CRC 4 |
| | | CRC, ECC Bytes | CRC 6 | CRC 7 | ECC 1[0] | ECC 2[0] | ECC 0[0] | ECC 1[1] | ECC 2[1] |
| | | | | | | | | | ECC 0[1] |

Figure 4-84 ↑↑Flit Interleaving in a x8 Link §



| | | Optical Channel | | | |
|---|---|-----------------|----|----|----|
| | | 0 | 1 | 2 | 3 |
| 256 Bytes | 0 | 0 | 2 | 4 | 6 |
| | 1 | 8 | 16 | 24 | 1 |
| | 2 | 32 | 40 | 33 | 41 |
| | 3 | 48 | 56 | 49 | 57 |
| Mapping continues until entire Flit is mapped to 4 channels | | | | | |

Figure 4-86 Flit Interleaving of a x8 Link in a 2:1 Electrical Lane to Optical Channel Mapping

4.3.7.1 Forwarding Type Rules

A Retimer must determine what type of Symbols it is forwarding. The rules for inferring Electrical Idle are a function of the type of Symbols the Retimer is forwarding. If a Path forwards two consecutive TS0, TS1, or TS2 Ordered Sets, on any Lane, then the Path is forwarding training sets. ~~If In Non-Flit Mode, if a Path forwards eight consecutive Symbol Times of Idle data on all Lanes that are forwarding Symbols, then the Path is forwarding non-training sets.~~ When a Retimer transitions from forwarding training sets to forwarding non-training sets, the variable RT_error_data_rate is set to 2.5 GT/s if 8b/10b or 128b/130b encoding is being used and set to 32.0 GT/s if 1b/1b encoding is being used. ~~determination in Flit Mode is described in § Section 4.3.7.2 (where the RT_flit_mode_enabled variable is described).~~

ECN: Base 6.3
Optical△↔

4.3.7.2 Orientation, Lane Numbers, and Data Stream Mode Rules

The Retimer must determine the Port orientation, Lane assignment, Lane polarity, and Data stream Mode dynamically as the Link trains.

- When RT_LinkUp =0, the first Pseudo Port to receive two consecutive TS1 Ordered Sets with a non-PAD Lane number on any Lane, has its RT_port_orientation variable set to Upstream Port, and the other Pseudo Port has its RT_port_orientation variable set to Downstream Port.
 - ~~When an Optical Aware Retimer is deployed, the above requirement applies to the first Pseudo Port connected to an Electrical Link Segment (instead of simply the first Pseudo Port) to receive two consecutive TS1 Ordered Sets with a non-PAD Lane number on any Lane (its RT_port_orientation variable must be set to Upstream Port). These TS1 Ordered Sets will be transferred, over the optical channels, to the Pseudo Port in the Optical Aware Retimer connected to the other Electrical Link Segment. The Pseudo Port that receives the TS1 Ordered Sets from the optical side of the Link will have its RT_port_orientation variable set to Downstream Port.~~

ECN: Base 6.3
Optical△↔

- The Retimer plays no active part in Lane number determination. The Retimer must capture the Lane numbers with the RT_captured_lane_number variable at the end of the Configuration state, between the Link Components. This applies on the first time through Configuration, i.e., when the RT_LinkUp variable is set to 0b. Subsequent trips through Configuration during Link width configure must not change the Lane numbers. Lane numbers are required for the scrambling seed when using 128b/130b or 1b/1b. Link numbers are required in some cases when the Retimer is in Execution mode. Link numbers and Lane numbers are captured with the RT_captured_lane_number, and RT_captured_link_number variables whenever the first two consecutive TS2 Ordered Sets that contain non-PAD Lane and non-PAD Link numbers are received after the RT_LinkUp variable is set to 0b. A Retimer must function normally if Lane reversal occurs. When the Retimer has captured the Lane numbers and Link numbers the RT_LinkUp variable is set to 1b. In addition, if the Disable Scrambling bit in the TS2 Ordered Sets is set to 1b, in either case above, then the Retimer determines that scrambling is disabled when using 8b/10b encoding.
- Lane polarity is determined any time the Lane exits Electrical Idle, and achieves Symbol lock at 2.5 GT/s as described in § Section 4.2.5.5 :
 - If polarity inversion is determined the Receiver must invert the received data. The Transmitter must never invert the transmitted data.
- The Retimer plays an active part of Data Stream Mode determination. If the Retimer supports Flit Mode operation, for each Pseudo Port, it must capture the value of the Flit Mode Supported bit of the Data Rate Identifier field in the eight consecutive TS2 Ordered Sets received with Link and Lane numbers set to PAD when the RT_LinkUp variable is set to 0b. If the Flit Mode Supported bit is 1b in eight consecutive TS2 Ordered Sets received by both Pseudo Ports, the RT_flit_mode_enabled variable must be set to 1b and each Pseudo Port must follow Flit Mode rules (as specified in § Section 4.2) to identify transitions between Ordered Set Data and Data Streams. If the Retimer does not support Flit Mode operation, its RT_flit_mode_enabled variable must remain set to 0b and it must set bit 0 of the Data Rate Identifier (Symbol 0) to 0b in all TS Ordered Sets that it forwards (as described in § Section 4.3.7.7).
 - When using 8b/10b with Flit Mode, NOP Flits (instead of Idle data) identify the start of the data stream.
 - When using 128b/130b or 1b/1b with Flit Mode, SDS Ordered Sets identify the start of the data stream.
 - ↑↑When an Optical Retimer Solution is deployed, the RT_flit_mode_enabled variable must be set to 1b when both the Upstream Pseudo Port connected to the Electrical Link Segment and the Downstream Pseudo Port connected to the other Electrical Link Segment each receive TS2 Ordered Sets with Link and Lane Numbers set to PAD, and the Flit Mode Supported bit set to 1b. Each Optical Aware Retimer must communicate the successful reception of the TS2 Ordered Sets to the companion Optical Aware Retimer (over the optical channel) so that each Retimer can set its RT_flit_mode_enabled variable to 1b. If the conditions to do so are not met, behavior is undefined. If an implementation with other than 2 optical based Retimers is deployed, the details of setting the RT_flit_mode_enabled variable are implementation specific.↑
- The Retimer's place in the system topology is determined when eight consecutive TS2 Ordered Sets are received with (non-PAD) matching Link and Lane numbers and identical data rate identifiers. If the Retimer Present bits are set to 01b on the Upstream Pseudo Port, the RT_number must be set to 10b, otherwise RT_number must be set to 01b. The RT_number is used to determine Pseudo Ports B, C, D and E. This identification is needed for Execution Mode Follower Loopback with 1b/1b encoding.
↑↑When an Optical Retimer Solution is deployed, the Upstream Pseudo Port connected to the Electrical Link Segment is RT_number 01b and Pseudo Port B, the Downstream Pseudo Port connected to the Electrical Link Segment is RT_number 10b and Pseudo Port E.↑

ECN: Base 6.3
Optical△ECN: Base 6.3
Optical△

4.3.7.3 Electrical Idle Exit Rules §

At data rates other than 2.5 GT/s, EIEOS are sent within the training sets to ensure that the analog circuit detects an exit from Electrical Idle. Receiving an EIEOS is required when using 128b/130b or 1b/1b encoding to achieve Block Alignment. When the Retimer starts forwarding data after detecting an Electrical Idle exit, the Retimer starts transmitting on a training set boundary. The first training sets it forwards must be an EIEOS, when operating at data rates higher than 2.5 GT/s. The first EIEOS sent will be in place of the TS0, TS1, or TS2 Ordered Set that it would otherwise forward.

~~When an Optical Retimer Solution is deployed, the training sets forwarded by the Downstream Pseudo Port were received by the Upstream Pseudo Port and transferred over the optical channel and the training sets forwarded by the Upstream Port were received by the Downstream Port and transferred over the optical channel.~~

ECN: Base 6.3
Optical

If no Lanes ~~on a Pseudo Port~~ meet Z RX-DC ~~on a Pseudo Port, and~~:

Errata: Base 6.3
B816

- ~~And, if~~ the following sequence occurs:
 - An exit from Electrical Idle is detected on any Lane of that Pseudo Port.
 - And then if not all Lanes infer Electrical Idle, via absence of exit from Electrical Idle in a 12 ms window on that Pseudo Port and the other Pseudo Port is not receiving Ordered Sets on any Lane in that same 12 ms window.
- Then ~~the same~~ ~~that~~ Pseudo ~~Port, where no Lanes meet Z RX-DC,~~ ~~Port~~ sends the Electrical Idle Exit pattern described below for 5 µs on all Lanes.

If operating at ~~2.5 GT/s and~~ ~~2.5 GT/s:~~

Errata: Base 6.3
B816

- ~~And, if~~ the following occurs:
 - any Lane detects an exit from Electrical Idle
 - and then receives two consecutive TS1 Ordered Sets with Lane and Link numbers equal to PAD
 - and the other Pseudo Port is not receiving Ordered Sets on any Lane
- Then Receiver Detection is performed on all Lanes of the Pseudo Port that is not receiving Ordered Sets. If no Receivers were detected then:
 - The result is back propagated as described in § Section 4.3.4, within 1.0 ms.
 - The same Pseudo Port that received the TS1 Ordered Sets with Lane and Link numbers equal to PAD, sends the Electrical Idle Exit pattern described below for 5 µs on all Lanes.

If a Lane detects an exit from Electrical ~~Idle then~~ ~~Idle:~~

Errata: Base 6.3
B816

- ~~Then~~ the Lane must start forwarding when all ~~of~~ the following are true:
 - ~~The Lane has detected a Receiver on the other Pseudo Port as described in § Section 4.3.4.~~
 - Data rate is determined, see § Section 4.3.7.4, current data rate is changed to RT_next_data_rate if required.
 - Lane polarity is determined, see § Section 4.3.7.2.
 - Two consecutive TS0, TS1, or TS2 Ordered Sets are received.

- Two consecutive TS0, TS1, or TS2 Ordered Sets are received on all Lanes that detected an exit from Electrical Idle or the max Retimer Exit Latency has occurred, see § Table 4-76.
 - Lane De-skew is achieved on all Lanes that received two consecutive TS0, TS1, or TS2 Ordered Sets.
 - If a data rate change has occurred then 6 µs has elapsed since Electrical Idle Exit was detected.
 - All Ordered Sets used to establish forwarding must be discarded. Only Lanes that have detected a Receiver on the other Pseudo Port, as described in § Section 4.3.4, are considered for forwarding.

↑↑When an Optical Retimer Solution is deployed, the “other Pseudo Port” refers to the Pseudo Port in the other Optical Aware Retimer that is connected to the Electrical Link Segment.↑

Errata: Base 6.3
B816△◀
- Otherwise after a 3.0 ms timeout, if the other Pseudo Port is not receiving Ordered Sets then Receiver Detection is performed on all Lanes of the Pseudo Port that is not receiving Ordered Sets, the result is back propagated as described in § Section 4.3.4, and if no Receivers were detected:
- Then the same Pseudo Port that was unable to receive two consecutive TS0, TS1, or TS2 Ordered Sets on any Lane sends the Electrical Idle Exit pattern described below for 5 µs on all Lanes.
 - Else the Electrical Idle Exit pattern described below is forwarded on all Lanes that detected an exit from Electrical Idle.
 - When using 8b/10b encoding:
 - The Modified Compliance Pattern with the error status Symbol set to 00h.
 - When using 128b/130b encoding:
 - One EIEOS or one EIEOSQ (recommended).
 - 32 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled, for Symbols 0 to 13.
 - Symbol 14 and 15 of each Data Block either contain Idle data Symbols (00h), scrambled, or DC Balance, determined by applying the same rules in § Section 4.2.5.1 to these Data Blocks.
 - When using 1b/1b encoding:
 - Four consecutive EIEOS, uninterrupted by any other Ordered Set including the Control SKP Ordered Set.
 - 32 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled, for Symbols 0-6, 8-14.
 - Symbol 7 and 15 of each Data Block either contain Idle data Symbols (00h), scrambled, or DC Balance, determined by applying the same rules in § Section 4.2.5.1 to these Data Blocks.
 - This Path now is forwarding the Electrical Idle Exit pattern. In this state Electrical Idle is inferred by the absence of Electrical Idle Exit, see § Table 4-77. The Path continues forwarding the Electrical Idle Exit pattern until Electrical Idle is inferred on any lane, or a 48 ms time out occurs. If a 48 ms time out occurs then:
 - The RT_LinkUp variable is set to 0b.
 - The Pseudo Port places its Transmitter in Electrical Idle
 - The RT_next_data_rate and the RT_error_data_rate must be set to 2.5 GT/s for both Pseudo Ports
 - Receiver Detection is performed on the Pseudo Port that was sending the Electrical Idle Exit pattern and timed out, the result is back propagated as described in § Section 4.3.4.
 - The Transmitter, on the opposite Pseudo Port that was sending the Electrical Idle Exit Pattern and timed out, sends the Electrical Idle Exit Pattern described above for 5 µs.

IMPLEMENTATION NOTE: ELECTRICAL IDLE EXIT §

Forwarding Electrical Idle Exit occurs in error cases where a Retimer is unable to decode training sets. Upstream and Downstream Ports use Electrical Idle Exit (without decoding any Symbols) during Polling.Compliance, and Recovery.Speed. If the Retimer does not forward Electrical Idle Exit then the Upstream and Downstream Ports will misbehave in certain conditions. For example, this may occur after a speed change to a higher data rate. In this event forwarding Electrical Idle Exit is required to keep the Upstream and Downstream Ports in lock step at Recovery.Speed, so that the data rate will return to the previous data rate, rather than a Link Down condition from a time out to Detect.

When a Retimer detects an exit from Electrical Idle and starts forwarding data, the time this takes is called the Retimer Exit Latency, and allows for such things as data rate change (if required), clock and data recovery, Symbol lock, Block Alignment, Lane-to-Lane de-skew, Receiver tuning, etc. The Maximum Retimer Exit Latency is specified below for several conditions:

- The data rate before and after Electrical Idle and Electrical Idle exit detect does not change.
- Data rate change to a data rate that uses 8b/10b encoding.
- Data rate change to a data rate that uses 128b/130b encoding for the first time.
- Data rate change to a data rate that uses 128b/130b encoding not for the first time.
- Data rate change to a data rate that uses 1b/1b encoding for the first time.
- Data rate change to a data rate that uses 1b/1b encoding not for the first time.
- How long both transmitters have been in Electrical Idle when a data rate change occurs.

Retimers are permitted to change their data rate while in Electrical Idle, and it is recommended that Retimers start the data rate change while in Electrical Idle to minimize Retimer Exit latency.

Table 4-76 Maximum Retimer Exit Latency §

| Condition | Link in Electrical Idle for X µs, where: | |
|--|--|------------|
| | X < 500 µs | X ≥ 500 µs |
| No data rate change, 2.5 GT/s | 8 µs | 8 µs |
| No data rate change, 5.0 GT/s or higher | 4 µs | 4 µs |
| When forwarding TS1 Ordered Sets at 2.5 GT/s with Lane and Link number equal to PAD. | 1 ms | 1 ms |
| Any data rate change to 8b/10b encoding data rate | 504 – X µs | 4 µs |
| First data rate change to 128b/130b encoding date rate | 1.5 – X ms | 1 ms |
| Subsequent data rate change to 128b/130b encoding date rate | 504 – X µs | 4 µs |
| First data rate change to 1b/1b encoding data rate | 1.5 – X ms | 1 ms |
| Subsequent data rate change to 1b/1b encoding data rate | 504 – X µs | 4 µs |

4.3.7.4 Data Rate Change and Determination Rules §

The data rate of the Retimer is set to 2.5 GT/s after deassertion of Fundamental Reset.

Both Pseudo Ports of the Retimer must operate at the same data rate. If a Pseudo Port places its Transmitter in Electrical Idle, then the Symbols that it has just completed transmitting determine the variables RT_next_data_rate and RT_error_data_rate. Only when both Pseudo Ports have all Lanes in Electrical Idle shall the Retimer change the data rate. If both Pseudo Ports do not make the same determination of these variables, then both variables must be set to 2.5 GT/s.

- If both Pseudo Ports were forwarding non-training sequences, then the RT_next_data_rate must be set to the current data rate. The RT_error_data_rate must be set to 2.5 GT/s if 8b/10b or 128b/130b encoding is being used. The RT_error_data_rate is set to 32.0 GT/s if 1b/1b encoding is being used. Note: this covers the case where the Link has entered L1 from L0.
- If both Pseudo Ports were forwarding TS2 Ordered Sets with the speed_change bit set to 1b and either:
 - the data rate, when forwarding those TS2s, is greater than 2.5 GT/s or,
 - the highest common data rate received in the data rate identifiers in both directions is greater than 2.5 GT/s,
 then RT_next_data_rate must be set to the highest common data rate and the RT_error_data_rate is set to current data rate. Note: this covers the case where the Link has entered Recovery.Speed from Recovery.RcvrCfg and is changing the data rate according to the highest common data rate.
- Else the RT_next_data_rate must be set to the RT_error_data_rate. The RT_error_data_rate is set to 2.5 GT/s if 8b/10b or 128b/130b encoding is being used. The RT_error_data_rate is set to 32.0 GT/s if 1b/1b encoding is being used. Note this covers the two error cases:
 - This indicates that the Link was unable to operate at the current data rate (greater than 2.5 GT/s) and the Link will operate at either the 2.5 GT/s data rate or the 32.0 GT/s data rate or
 - This indicates that the Link was unable to operate at the new negotiated data rate and will revert back to the old data rate with which it entered Recovery from L0 or L1.

↑↓When an Optical Retimer Solution is being deployed, the above rules apply to the Upstream Pseudo Port connected to one Electrical Link Segment and the Downstream Pseudo Port connected to the other Electrical Link Segment (i.e., these are the Ports that comprise “both Pseudo Ports” in topologies using Optical Aware Retimers). The RT_error_data_rate and RT_next_data_rate variables must be set identically in both Retimers. The communication of the information used to determine RT_error_data_rate and RT_next_data_rate between the Optical Aware Retimers (across the optical channel), is performed in an implementation specific manner.↑

ECN: Base 6.3
Optical△↔

4.3.7.5 Electrical Idle Entry Rules §

The Rules for Electrical Idle entry in Forwarding mode are a function of whether the Retimer is forwarding training sets or non-training sets. The determination of this is described in § Section 4.3.7.1.

Before a Transmitter enters Electrical Idle, it must always send the Electrical Idle Ordered Set Sequence (EIOSQ), unless otherwise specified.

If the Retimer is forwarding training sets then:

- If an EIOS is received on a Lane, then the EIOSQ is forwarded on that Lane and only that Lane places its Transmitter in Electrical Idle.

- If Electrical Idle is inferred on a Lane, then that Lane places its Transmitter in Electrical Idle, after EIOSQ is transmitted on that Lane.
- In both of the above cases, when an Optical Retimer Solution is being deployed, the EIOSQ must be transmitted by the associated Lane of the Pseudo Port connected to the Electrical Link Segment on the other side of the optical extension. That Transmitter is then placed into Electrical Idle. The communication of these conditions (either receiving an EIOS or inferring Electrical Idle) across the optical channel is performed in an implementation specific manner.

ECN: Base 6.3
Optical $\triangle\triangle$

Else if the Retimer is forwarding non-training sets then:

- If operating in Flit Mode and EIOS are received on some Lanes while some other Lanes receive SKP OS (i.e., L0p Link width down-size), then Lanes that are receiving EIOS must forward the EIOSQ and must place their Transmitters into Electrical Idle. Lanes that are forwarding Symbols, but are not receiving EIOS, must continue forwarding Symbols and must not place their Transmitters into Electrical Idle. Else if an EIOS is received on any Lane, then the EIOSQ is forwarded on all Lanes that are currently forwarding Symbols and all Lanes place their Transmitters in Electrical Idle.
- If Electrical Idle is inferred on a Lane, then that Lane places its Transmitter in Electrical Idle, and EIOSQ is not transmitted on that Lane.
- When operating at 64.0 GT/s, a Retimer must follow the requirements of § Section 4.2.3.2 in order to identify SKP OS, EIEOS, and EIOS received.
- When an Optical Retimer Solution is being deployed, and any of the above conditions are met, the resulting action must be taken on the Transmitters of the associated Lanes of the Pseudo Port connected to the Electrical Link Segment on the other side of the Link. The communication of these conditions across the optical channel is performed in an implementation specific manner.

ECN: Base 6.3
Optical $\triangle\triangle$

The Retimer is required to infer Electrical Idle. The criteria for a Retimer inferring Electrical Idle are described in § Table 4-77 .

Table 4-77 Inferring Electrical Idle §

| State | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s or higher |
|---|---|---|---|--|
| Forwarding: Non Training Sequence | Absence of a <u>SKP</u> <u>Ordered Set</u> in a <u>128 μs</u> window | Absence of a <u>SKP Ordered</u> <u>Set</u> in a <u>128 μs</u> window | Absence of a <u>SKP Ordered</u> <u>Set</u> in a <u>128 μs</u> window | Absence of a <u>SKP Ordered</u> <u>Set</u> in a <u>128 μs</u> window |
| Forwarding: Training Sequence | Absence of a <u>TS1</u> or <u>TS2</u> <u>Ordered Set</u> in a <u>1280 UI</u> interval | Absence of a <u>TS1</u> or <u>TS2</u> <u>Ordered Set</u> in a <u>1280 UI</u> interval | Absence of a <u>TS1</u> or <u>TS2</u> <u>Ordered Set</u> in a <u>4680 UI</u> interval | Absence of a <u>TS0</u> , <u>TS1</u> , or <u>TS2</u> <u>Ordered Set</u> in a <u>4680 UI</u> interval |
| Forwarding: Electrical Idle Exit | Absence of an exit from Electrical Idle in a 2000 UI interval | Absence of an exit from Electrical Idle in a 16000 UI interval | Absence of an exit from Electrical Idle in a 16000 UI interval | Absence of an exit from Electrical Idle in a 16000 UI interval |
| Executing: Force Timeout | | | | |
| Forwarding: Loopback | Absence of an exit from Electrical Idle in a <u>128 μs</u> window | N/A | N/A | N/A |
| Executing: Loopback Follower | | | | |

4.3.7.6 Transmitter Settings Determination Rules §

When a data rate change to 64.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- If the RT_G6_EQ_complete variable is set to 1b:
 - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure applicable to 64.0 GT/s operation.
- Else:
 - An Upstream Pseudo Port must use the 128b/130b Transmitter preset values it registered from the eight consecutive 128b/130b EQ TS2 Ordered Sets received while operating at 32.0 GT/s in its Transmitter preset setting as soon as it starts transmitting at the 64.0 GT/s data rate and must ensure that it meets the preset definition in § Section 4.2.4.2 . Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it starts transmitting at 64.0 GT/s.
 - A Downstream Pseudo Port determines its Transmitter Settings in an implementation specific manner when it starts transmitting at 64.0 GT/s.

The RT_G6_EQ_complete variable is set to 1b when:

- Two consecutive TS0 Ordered Sets are received with EC = 01b at 64.0 GT/s.

The RT_G6_EQ_complete variable is set to 0b when any of the following occur:

- The RT_LinkUp variable is set to 0b.
- The Pseudo Port is operating at 32.0 GT/s and eight consecutive 128b/130b EQ TS2 Ordered Sets are received on any Lane of the Upstream Pseudo Port. The value in the 128b/130b Transmitter Preset field is registered for later use at 64.0 GT/s for that Lane. ↑↑When an Optical Retimer Solution is being deployed and the conditions for setting the RT_G6_EQ_complete variable to 0b are met, the Upstream Port must communicate the requirement to the Downstream Port in an implementation specific manner.↑

ECN: Base 6.3
Optical

When a data rate change to 32.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- If the RT_G5_EQ_complete variable is set to 1b:
 - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure applicable to 32.0 GT/s operation.
- Else:
 - An Upstream Pseudo Port must use the 128b/130b Transmitter preset values it registered from the eight consecutive 128b/130b EQ TS2 Ordered Sets received while operating at 16.0 GT/s in its Transmitter preset setting as soon as it starts transmitting at the 32.0 GT/s data rate and must ensure that it meets the preset definition in § Section 4.2.4.2 . Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it starts transmitting at 32.0 GT/s.
 - A Downstream Pseudo Port determines its Transmitter Settings in an implementation specific manner when it starts transmitting at 32.0 GT/s.

The RT_G5_EQ_complete variable is set to 1b when:

- Two consecutive TS1 Ordered Sets are received with EC = 01b at 32.0 GT/s.

The RT_G5_EQ_complete variable is set to 0b when any of the following occur:

- RT_LinkUp variable is set to 0b.
- The Pseudo Port is operating at 16.0 GT/s and eight consecutive 128b/130b EQ TS2 Ordered Sets are received on any Lane of the Upstream Pseudo Port. The value in the 128b/130b Transmitter Preset field is registered for later use at 32.0 GT/s for that Lane.
 - When an Optical Retimer Solution is being deployed, either Bypass Equalization to the Highest NRZ rate or No Equalization Needed must be supported and 16.0 GT/s operation is not required. When Bypass Equalization to the Highest NRZ rate is negotiated, the above requirement for 128b/130b EQ TS2 is for EQ TS2 (because those Ordered Sets will be received at an 8b/10b rate instead of 128b/130b). When the conditions for setting the RT_G5_EQ_complete variable to 0b are met, the Upstream Port must communicate the requirement to the Downstream Port in an implementation specific manner.

 ECN: Base 6.3
 Optical△↔

When a data rate change to 16.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- If the RT_G4_EQ_complete variable is set to 1b:
 - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure applicable to 16.0 GT/s operation.
- Else:
 - An Upstream Pseudo Port must use the 128b/130b Transmitter preset values it registered from the received eight consecutive 128b/130b EQ TS2 Ordered Sets in its Transmitter preset setting as soon as it starts transmitting at the 16.0 GT/s data rate and must ensure that it meets the preset definition in § Section 8.3.3.3 . Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it starts transmitting at 16.0 GT/s.
 - A Downstream Pseudo Port determines its Transmitter Settings in an implementation specific manner when it starts transmitting at 16.0 GT/s.

The RT_G4_EQ_complete variable is set to 1b when:

- Two consecutive TS1 Ordered Sets are received with EC = 01b at 16.0 GT/s.

The RT_G4_EQ_complete variable is set to 0b when any of the following occur:

- The RT_LinkUp variable is set to 0b.
- Eight consecutive 128b/130b EQ TS2 Ordered Sets are received on any Lane of the Upstream Pseudo Port. The value in the 128b/130b Transmitter Preset field is registered for later use at 16.0 GT/s for that Lane.

When a data rate change to 8.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- If the RT_G3_EQ_complete variable is set to 1b:
 - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure applicable to 8.0 GT/s operation.
- Else:
 - An Upstream Pseudo Port must use the 8.0 GT/s Transmitter preset values it registered from the received eight consecutive EQ TS2 Ordered Sets in its Transmitter preset setting as soon as it starts transmitting at the 8.0 GT/s data rate and must ensure that it meets the preset definition in § Section 8.3.3 . Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it

starts transmitting at 8.0 GT/s. The Upstream Pseudo Port may optionally use the 8.0 GT/s Receiver preset hint values it registered in those EQ TS2 Ordered Sets.

- A Downstream Pseudo Port determines its Transmitter preset settings in an implementation specific manner when it starts transmitting at 8.0 GT/s.

The RT_G3_EQ_complete variable is set to 1b when:

- Two consecutive TS1 Ordered Sets are received with EC = 01b at 8.0 GT/s.

The RT_G3_EQ_complete variable is set to 0b when any of the following occur:

- The RT_LinkUp variable is set to 0b.
- Eight consecutive EQ TS1 or eight consecutive EQ TS2 Ordered Sets are received on any Lane of the Upstream Pseudo Port. The value in the 8.0 GT/s Transmitter Preset and optionally the 8.0 GT/s Receiver Preset Hint fields are registered for later use at 8.0 GT/s for that Lane.

When a data rate change to 5.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- The Upstream Pseudo Port must sets its Transmitters to either -3.5 dB or -6.0 dB, according to the Selectable De-emphasis bit (bit 6 of Symbol 4) received in eight consecutive TS2 Ordered Sets, in the most recent series of TS2 Ordered sets, received prior to entering Electrical Idle.
- The Downstream Pseudo Port sets its Transmitters to either -3.5 dB or -6.0 dB in an implementation specific manner.

4.3.7.7 Ordered Set Modification Rules §

Ordered Sets are forwarded, and certain fields are modified according to the following rules:

- The Retimer shall not modify any fields except those specifically allowed/required for modification in this specification.
- Transmitter Precode Request: the Retimer shall overwrite the Transmitter Precode Request field in TS1 and TS2 Ordered Sets in both directions. The new value represents whether one pseudo port is requesting to enable precoding for the current data rate.
- Transmitter Precoding On: the Retimer shall overwrite Transmitter Precoding On field in TS1 Ordered Sets in both directions. The new value represents whether one pseudo port's precoding is on for the current data rate.
- LF: the Retimer shall overwrite the LF field in TS0 / TS1 Ordered Sets transmitted in both directions. The new value is determined in an implementation specific manner by the Retimer.
- FS: the Retimer shall overwrite the FS field in TS0 / TS1 Ordered Sets transmitted in both directions. The new value is determined in an implementation specific manner by the Retimer.
- Respective Pre-Cursor Coefficients: the Retimer shall overwrite the respective Pre-Cursor Coefficient field in TS0 / TS1 Ordered Sets transmitted in both directions. The new value is determined by the current Transmitter settings.
- Cursor Coefficient: the Retimer shall overwrite the Cursor Coefficient field in TS0 / TS1 Ordered Sets transmitted in both directions. The new value is determined by the current Transmitter settings.
- Post-Cursor Coefficient: the Retimer shall overwrite the Post-Cursor Coefficient field in the TS0 / TS1 Ordered Sets transmitted in both directions. The new value is determined by the current Transmitter settings.
- Parity: the Retimer shall overwrite the Parity bit of forwarded TS0 , TS1 , TS2 , or Modified TS1/TS2 Ordered Sets if it modifies any field used in parity calculation.

- Transmitter Preset: the Retimer shall overwrite the Transmitter Preset field in TS0 / TS1 Ordered Sets transmitted in both directions. If the Transmitter is using a Transmitter preset setting then the value is equal to the current setting, else it is recommended that the Transmitter Preset field be set to the most recent Transmitter preset setting that was used for the current data rate.

The Retimer is permitted to do the following:

- overwrite the Transmitter Preset field in EQ TS1 Ordered Sets in either direction
- overwrite the 8.0 GT/s Transmitter Preset field in EQ TS2 Ordered Sets in the Downstream direction.
- overwrite the 128b/130b Transmitter Preset field in 128b/130b EQ TS2 Ordered Sets, in the Downstream direction.

The new values for the 8.0 GT/s Transmitter Preset and 128b/130b Transmitter Preset fields are determined in an implementation specific manner by the Retimer.

During phase ~~↓↑0 of ↓~~ ~~↓↑1 of ↑~~ Equalization to 16.0 GT/s (i.e., the current Data Rate is 8.0 GT/s), phase 0 of Equalization to 32.0 GT/s (i.e., the current Data Rate is 16.0 GT/s), or phase 0 of Equalization to 64.0 GT/s (i.e., the current Data Rate is 32.0 GT/s) the Retimer is permitted to do the following in the Upstream direction:

- Forward received TS2 Ordered Sets.
- Convert TS2 Ordered Sets to 128b/130b EQ TS2 Ordered Sets, the value for the 128b/130b Transmitter Preset field is determined in an implementation specific manner by the Retimer.
- Forward received 128b/130b EQ TS2 Ordered Sets with modification, the value for the 128b/130b Transmitter Preset field is determined in an implementation specific manner by the Retimer.
- Convert 128b/130b EQ TS2 Ordered Sets to TS2 Ordered Sets.
- ~~↓↑When an Optical Retimer Solution is being deployed, the Ordered Sets being forwarded, converted and/or modified in phase 0 of Equalization were received by the Downstream Pseudo Port connected to one Electrical Link Segment and transferred via the optical technology to the Upstream Pseudo Port connected to the other Electrical Link Segment↑~~

ECN: Base 6.3
Optical $\triangleleft\triangleright$

- Receiver Preset Hint: the Retimer is permitted to do the following:
 - overwrite the Receiver Preset Hint field in EQ TS1 Ordered Sets in either direction
 - overwrite the 8.0 GT/s Receiver Preset Hint field in EQ TS2 Ordered Sets in the Downstream direction.
 The new values, for the Receiver Preset Hint and 8.0 GT/s Receiver Preset Hint fields are determined in an implementation specific manner by the Retimer.
- SKP Ordered Set: The Retimer is permitted to adjust the length of SKP Ordered Sets transmitted in both directions. The Retimer must perform the same adjustment on all Lanes. When operating with 8b/10b encoding, the Retimer is permitted to add or remove one SKP Symbol of a SKP Ordered Set. When operating with 128b/130b encoding, a Retimer is permitted to add or remove 4 SKP Symbols of a SKP Ordered Set. When operating with 1b/1b encoding, only Control SKP Ordered Sets are sent.
- Control SKP Ordered Set : The Retimer must modify the First Retimer Data Parity, or the Second Retimer Data Parity, of the Control SKP Ordered Set when the Retimer is in forwarding mode at 16.0 GT/s or above, according to its received parity. The received even parity is computed independently on each Lane as follows:
 - Parity is initialized when a data rate change occurs.
 - Parity is initialized when a SDS Ordered Set is received.
 - Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed.
 - Parity is initialized when a Control SKP Ordered Set is received. However, parity is NOT initialized when a Standard SKP Ordered Set is received.

If a Pseudo Port detects the Retimer Present bit was 0b in the most recently received two consecutive TS2 or EQ TS2 Ordered Sets received by that Pseudo Port when operating at 2.5 GT/s then that Pseudo Port Receiver modifies the First Retimer Data Parity as it forwards the Control SKP Ordered Set , else that Pseudo Port Receiver modifies the Second Retimer Data Parity as it forwards the Control SKP Ordered Set .

The Retimer must modify symbols 4^*N+1 , 4^*N+2 , and 4^*N+3 of the Control SKP Ordered Set in the Upstream direction as described in § Section 4.2.18 $\uparrow\downarrow\downarrow$ $\uparrow\uparrow$ when Usage Model is 0b. When Usage Model is 1b, those symbols contain sideband information. The fields may be modified or forwarded unmodified depending on their intended usage, which is implementation specific. \uparrow

ECN: Base 6.3
Optical $\triangle\triangle$

See § Section 4.2.8.2 for Control SKP Ordered Set definition.

- When operating with 1b/1b encoding, a Retimer is permitted to add or remove 8 Bytes of SKP at an aligned 8 byte boundary.
- Selectable De-emphasis: the Retimer is permitted to overwrite the Selectable De-emphasis field in the TS1 or TS2 Ordered Set in both directions. The new value is determined in an implementation specific manner by the Retimer.
- The Data Rate Identifier: The Retimer must set the Data Rate Supported bits of the Data Rate Identifier Symbol consistent with the data rates advertised in the received Ordered Sets and its own max supported Data Rate, i.e., it clears to 0b all Symbol 4 bits[5:0] Data Rates that it does not support. A Retimer must support all data rates below and including its maximum supported data rate. A Retimer makes its determination of maximum supported Data Rate once, after fundamental reset. A Retimer that does not support Flit Mode must set Symbol 4, bit 0 to 0b.
 - $\uparrow\uparrow$ Retimers using optical technology (Optical Aware Retimers) must support Flit Mode \uparrow
- DC Balance: When operating with 128b/130b and 1b/1b encoding, the Retimer tracks the DC Balance of its Pseudo Port transmitters and transmits DC Balance Symbols as described in § Section 4.2.5.1.
- Retimer Present: When operating at 2.5 GT/s, the Retimer must Set the Retimer Present bit in all forwarded Ordered Sets with a defined Retimer Present bit. $\uparrow\uparrow$ Optical Aware Retimers must set this bit to 0b. \uparrow
- Two Retimers Present: If the Retimer supports 16.0 GT/s or higher, then when operating at 2.5 GT/s, the Retimer must Set the Two Retimers Present bit in all forwarded Ordered Sets with a defined Two Retimers Present bit, if it receives an Ordered Set that has a defined Retimer Present bit and the Retimer Present bit is Set. If the Retimer does not support 16.0 GT/s or higher, then when operating at 2.5 GT/s, the Retimer is permitted to Set the Two Retimers Present bit of all forwarded Ordered Sets with a defined Two Retimers Present bit, if it receives an Ordered Set that has a defined Retimer Present bit and the Retimer Present bit is Set.
- $\uparrow\uparrow$ N_FTS[7:6]: When the RT_flt_mode_enabled variable is set to 1b, Optical Aware Retimers must set these bits to 10b (Optical Aware Retimer Solution Present). Note that an earlier version of this specification permitted that N_FTS[7:6] be allowed to traverse a Retimer without modification, but this is now strongly discouraged for Optical Aware Retimer Solutions. \uparrow
- Loopback: When optionally supporting Follower Loopback in Execution mode Loopback_Request must be deasserted when forwarding training sets.
- Enhanced Link Behavior Control: If the Retimer supports 32.0 GT/s or higher, then when operating at 2.5GT/s, the Retimer must set the Enhanced Link Behavior Control bits of all forwarded TS1, TS2, EQ TS1 and EQ TS2 Ordered Sets as follows:

ECN: Base 6.3
Optical $\triangle\triangle$

- Set to 11b when Retimer supports Modified TS1/TS2 Ordered Sets and the Enhanced Link Behavior Control bits set to 11b in the Ordered Sets received for forwarding.
- Set to 10b when Retimer supports no equalization and the Enhanced Link Behavior Control bits is set to 10b in the Ordered Sets received for forwarding.
- Set to 01b when Retimer supports Equalization Bypass to Highest NRZ Rate and the Enhanced Link Behavior Control field is set to 01b in the Ordered Sets received for forwarding.
- Otherwise, set to 00b.

4.3.7.8 DLLP, TLP, Logical Idle, and Flit Modification Rules §

DLLPs, TLPs, Logical Idle, and Flits are forwarded with no modifications to any of the Symbols unless otherwise specified.

4.3.7.9 8b/10b Encoding Rules §

The Retimer shall meet the requirements in § Section 4.2.1.1.3 except as follows:

- When the Retimer is forwarding and an 8b/10b decode error or a disparity error is detected in the received data, the Symbol with an error is replaced with the D21.3 Symbol with incorrect disparity in the forwarded data.

IMPLEMENTATION NOTE: FORWARDING D21.3 SYMBOL WITH INCORRECT DISPARITY §

Detection of 8b/10b decode errors and disparity errors takes place in the Receiver. Whether replacement of the Symbol in error takes place in the Receiver or in the Transmitter, the Symbol forwarded by the Retimer after Scrambling will be the D21.3 Symbol with incorrect disparity.

- This clause in § Section 4.2.1.1.3 does not apply: If a received Symbol is found in the column corresponding to the incorrect running disparity or if the Symbol does not correspond to either column, the Physical Layer must notify the Data Link Layer that the received Symbol is invalid. This is a Receiver Error, and is a reported error associated with the Port (see § Section 6.2).

IMPLEMENTATION NOTE: RETIMER TRANSMITTER DISPARITY §

The Retimer must modify certain fields of the TS1 and TS2 Ordered Sets (e.g., Receiver Preset Hint, Transmitter Preset), therefore the Retimer must recalculate the running disparity. Simply using the disparity of the received Symbol may lead to an error in the running disparity. For example, some 8b/10b codes have 6 ones and 4 zeros for positive disparity, while other codes have 5 ones and 5 zeros.

4.3.7.10 8b/10b Scrambling Rules §

A Retimer is required to determine if scrambling is disabled when using 8b/10b encoding as described in § [Section 4.3.7.2](#).

4.3.7.11 Hot Reset Rules §

~~If any Lane of the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with Hot_Reset_Request asserted and then both Pseudo Ports either receive an EIOS or infer Electrical Idle on any Lane, that is receiving TS1 Ordered Sets, the Retimer does the following: What does "that is receiving TS1 Ordered Sets" mean? I think this paragraph should be rewritten:~~

If any Lane of the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with Hot_Reset_Request asserted followed by both Pseudo Ports either receiving an EIOS or inferring Electrical Idle on any ~~Lane, of those Lanes,~~ the Retimer does the following:

Errata: Base 6.3
B829△◀▷

- Clears variable RT_LinkUp = 0b.
- Places its Transmitters in Electrical Idle on both Pseudo Ports.
- Set the RT_next_data_rate variable to 2.5 GT/s.
- Set the RT_error_data_rate variable to 2.5 GT/s.
- Waits for an exit from Electrical Idle on any Lane on either Pseudo Port.
- ~~When an Optical Retimer Solution is being deployed, the Transmitters of the Upstream Pseudo Port connected to one Electrical Link Segment and the Transmitters of the Downstream Pseudo Port connected to the other Electrical Link Segment must be placed in Electrical Idle.~~

ECN: Base 6.3
Optical△◀▷

The Retimer does not perform Receiver Detection on either Pseudo Port.

4.3.7.12 Disable Link Rules §

~~If any Lane of the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with Disable_Link_Request asserted and then both Pseudo Ports either receive an EIOS or infer Electrical Idle on any Lane, that is receiving TS1 Ordered Sets, the Retimer does the following: What does "that is receiving TS1 Ordered Sets" mean? I think this paragraph should be rewritten:~~

If any Lane of the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with Disable_Link_Request asserted followed by both Pseudo Ports either receiving an EIOS or inferring Electrical Idle on any ~~Lane, of those Lanes,~~ the Retimer does the following:

Errata: Base 6.3
B829△◀▷

- Clears variable RT_LinkUp = 0b.
- Places its Transmitters in Electrical Idle on both Pseudo Ports.
- Set the RT_next_data_rate variable to 2.5 GT/s.
- Set the RT_error_data_rate variable to 2.5 GT/s.
- Waits for an exit from Electrical Idle on any Lane on either Pseudo Port.

- When an Optical Retimer Solution is being deployed, the Transmitters of the Upstream Pseudo Port connected to one Electrical Link Segment and the Transmitters of the Downstream Pseudo Port connected to the other Electrical Link Segment must be placed in Electrical Idle.¹

ECN: Base 6.3
Optical[△]

The Retimer does not perform Receiver Detection on either Pseudo Port.

4.3.7.13 Loopback §

The Retimer must operate in Follower Loopback Execution mode when operating at 8b/10b or 128b/130b encoding and any Lane receives two consecutive TS1 Ordered Sets with Loopback_Request asserted and the ability to execute Follower Loopback is configured in an implementation specific way.

The Retimer must operate in Follower Loopback Execution mode when operating at 1b/1b encoding and any Lane receives two consecutive TS1 Ordered Sets with Training Control bits [3:2] set to 01b (Assert Loopback) and with Training Control bits [1:0] matching the RT_number. See § Section 4.3.8.3 for Follower Loopback in Execution mode.

- When RT_number is 01b, the Retimer represents Pseudo-Ports B and C of the system topology (which are the ports being targeted for Follower Loopback Execution mode when Training Control = 0101b).
- When RT_number is 10b, the Retimer represents Pseudo-Ports D and E of the system topology (which are the ports being targeted for Follower Loopback Execution mode when Training Control = 0110b).

The Retimer follows these additional rules if one of the following conditions is met:

- Any Lane receives two consecutive TS1 Ordered Sets with Loopback_Request asserted and the ability to execute Follower Loopback is not configured in an implementation specific way.
- Retimer is operating at 1b/1b encoding and any Lane receives two consecutive TS1 Ordered Sets with Loopback_Request asserted and the conditions to enable Follower Loopback in Execution mode are not met. The setting does not configure Follower Loopback.

The purpose of these rules is to allow interoperation when a Retimer (or two Retimers) exist between a Loopback Lead and a Loopback Follower

- The Pseudo Port that received the TS1 Ordered Sets with Loopback_Request asserted acts as the Loopback Follower (the other Pseudo Port acts as Loopback Lead). The Upstream Path is defined as the Pseudo Port that is the Loopback Lead to the Pseudo Port that is the Loopback Follower. The other Path is the Downstream Path.
- Once established, if a Lane loses the ability to maintain Symbol Lock or Block alignment, then the Lane must continue to transmit Symbols while in this state.
- When using 8b/10b encoding and Symbol lock is lost, the Retimer must attempt to re-achieve Symbol Lock.
- When using 128b/130b encoding and Block Alignment is lost, the Retimer must attempt to re-achieve Block Alignment via SKP Ordered Sets.
- When using 1b/1b encoding and Block or Flit Alignment is lost, the Retimer must attempt to re-achieve Block and Flit Alignment via SKP Ordered Sets.
- If Loopback was entered while the Link Components were in Configuration.Linkwidth.Start, then determine the highest common data rate of the data rates supported by the Link via the data rates received in two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets on any Lane, that was receiving TS1 or TS2 Ordered Sets, at the time the transition to ForwardingLoopback occurred. If the current data rate is not the highest common data rate, then:
 - Wait for any Lane to receive EIOS, and then place the Transmitters in Electrical Idle for that Path.

- When all Transmitters are in Electrical Idle, adjust the data rate as previously determined.
- If the new data rate is 5.0 GT/s, then the Selectable De-emphasis is determined the same as way as described in § Section 4.2.7.10.1 .
- If the new data rate uses 128b/130b or 1b/1b encoding, then the Transmitter preset setting is determined the same as way as described in § Section 4.2.7.10.1 .
- In the Downstream Path; wait for Electrical Idle exit to be detected on each Lane and then start forwarding when two consecutive TS1 Ordered Sets have been received, on a Lane by Lane basis. This is considered the first time to this data rate for the Retimer Exit Latency.
- In the Upstream Path; if Compliance_Receive_Request in the TS1 Ordered Sets that directed the follower to this state was not asserted, then wait for Electrical Idle exit to be detected on each Lane, and start forwarding when two consecutive TS1 Ordered Sets have been received, on a Lane by Lane basis. This is considered the first time to this data rate for the Retimer Exit Latency.
- In the Upstream Path; if Compliance_Receive_Request in the TS1 Ordered Sets that directed the follower to this state was asserted, then wait for Electrical Idle exit to be detected on each Lane, and start forwarding immediately, on a Lane by Lane basis. This is considered the first time to this data rate for the Retimer Exit Latency.
- If four EIOS (one EIOS if the current data rate is 2.5 GT/s) are received on any Lane then:
 - Transmit eight EIOS on every Lane that is transmitting TS1 Ordered Sets on the Pseudo Port that did not receive the EIOS and place the Transmitters in Electrical Idle.
- When both Pseudo Ports have placed their Transmitters in Electrical Idle then:
 - Set the RT_next_data_rate variable to 2.5 GT/s.
 - Set the RT_error_data_rate variable to 2.5 GT/s.
 - The additional rules for Loopback no longer apply unless the rules for entering this section are met again.

4.3.7.14 Compliance Receive Rules §

The Retimer follows these additional rules if any Lane receives eight consecutive TS1 Ordered Sets (or their complement) with Compliance_Receive_Request asserted and Loopback_Request deasserted. The purpose of the following rules is to support Link operation with a Retimer when Compliance_Receive_Request is asserted and Loopback_Request is deasserted in TS1 Ordered Sets, transmitted by the Upstream or Downstream Port, while the Link is in Polling.Active.

- Pseudo Port A is defined as the first Pseudo Port that receives eight consecutive TS1 Ordered Sets (or their complement) with Compliance_Receive_Request asserted and Loopback_Request deasserted. Pseudo Port B is defined as the other Pseudo Port.
 - ↑When an Optical Retimer Solution is being deployed, Port B is the Pseudo Port in the other Optical Aware Retimer which is connected to the Electrical Link Segment. References to Pseudo Ports in this section represent the Pseudo Ports connected to the Electrical Link Segments at each end of the Optical Retimer Solution.↑
- The Retimer determines the highest common data rate of the Link by examining the data rate identifiers in the TS1 Ordered Sets received on each Pseudo Port, and the maximum data rate supported by the Retimer.
- If the highest common data rate is equal to 5.0 GT/s then:
 - The Retimer must change its data rate to 5.0 GT/s as described in § Section 4.3.7.4 .
 - The Retimer Pseudo Port A must set its de-emphasis according to the selectable de-emphasis bit received in the eight consecutive TS1 Ordered Sets.

ECN: Base 6.3
Optical△↔

- The Retimer Pseudo Port B must set its de-emphasis in an implementation specific manner.
- If the highest common data rate is equal to 8.0 GT/s or higher then:
 - The Retimer must change its data rate to as applicable, as described in § [Section 4.3.7.4](#).
 - Lane numbers are determined as described in § [Section 4.2.12](#).
 - The Retimer Pseudo Port A must set its Transmitter coefficients on each Lane to the Transmitter preset value advertised in Symbol 6 of the eight consecutive TS1 Ordered Sets and this value must be used by the Transmitter (use of the Receiver preset hint value advertised in those TS1 Ordered Sets is optional). If the common data rate is 8.0 GT/s or higher, any Lanes that did not receive eight consecutive TS1 Ordered Sets with Transmitter preset information can use any supported Transmitter preset setting in an implementation specific manner.
 - The Retimer Pseudo Port B must set its Transmitter and Receiver equalization in an implementation specific manner.
- The Retimer must forward the Modified Compliance Pattern when it has locked to the pattern. This occurs independently on each Lane in each direction. If a Lane's Receiver loses Symbol Lock or Block Alignment, the associated Transmitter (i.e., same Lane on opposite Pseudo Port) Continues to forward data.
- Once locked to the pattern, the Retimer keeps an internal count of received Symbol errors, on a per-Lane basis. The pattern lock and Lane error is permitted to be readable in an implementation specific manner, on a per-Lane basis.
- When operating with 128b/130b or 1b/1b encoding, Symbols with errors are forwarded unmodified by default, or may optionally be corrected to remove error pollution. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- When operating with 8b/10b encoding, Symbols with errors are replaced with the D21.3 Symbol with incorrect disparity by default, or may optionally be corrected to remove error pollution. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- The error status Symbol when using 8b/10b encoding or the Error_Status field when using 128b/130b or 1b/1b encoding is forwarded unmodified by default, or may optionally be redefined as it is transmitted by the Retimer. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- If any Lane receives an EIOS on either Pseudo Port then:
 - Transmit EIOSQ on every Lane of the Pseudo Port that did not receive EIOS and place the Transmitters in Electrical Idle. Place the Transmitters of the other Pseudo Port in Electrical Idle; EIOS is not transmitted by the other Pseudo Port.
 - Set the RT_next_data_rate variable to 2.5 GT/s.
 - Set the RT_error_data_rate variable to 2.5 GT/s.
 - The Compliance Receive additional rules no longer apply unless the rules for entering this section are met again.

4.3.7.15 Enter Compliance Rules §

The Retimer follows these additional rules if the Retimer is exiting Electrical Idle after entering Electrical Idle as a result of Hot Reset, and the Retimer Enter Compliance bit is Set in the Retimer. The purpose of the following rules is to support Link operation with a Retimer when the Link partners enter compliance as a result of the Enter Compliance bit in the Link Control 2 Register set to 1b in both Link Components and a Hot Reset occurring on the Link. Retimers do not support Link operation if the Link partners enter compliance when they exit detect if the entry into detect was not caused by a Hot Reset.

Retimers must support the following register fields in an implementation specific manner:

Base 6.4 vs Base 6.3

- Retimer Target Link Speed
 - One field per Retimer
 - Type = RWS
 - Size = 3 bits
 - Default = 001b
 - Encoding:
 - 001b = 2.5 GT/s
 - 010b = 5.0 GT/s
 - 011b = 8.0 GT/s
 - 100b = 16.0 GT/s
 - 101b = 32.0 GT/s
 - 110b = 64.0 GT/s
- Retimer Transmit Margin
 - One field per Pseudo Port
 - Type = RWS
 - Size = 3 bits
 - Default = 000b
 - Encoding:
 - 000b = Normal Operating Range
 - 001b-111b = As defined in § Section 8.3.4, not all encodings are required to be implemented
- Retimer Enter Compliance
 - One bit per Retimer
 - Type = RWS
 - Size = 1 bit
 - Default = 0b
 - Encoding:
 - 0b = do not enter compliance
 - 1b = enter compliance
- Retimer Enter Modified Compliance
 - One bit per Retimer
 - Type = RWS
 - Size = 1 bit
 - Default = 0b
 - Encoding:
 - 0b = do not enter modified compliance
 - 1b = enter modified compliance
- Retimer Compliance Preset/De-emphasis
 - One field per Pseudo Port

- Type = RWS
- Size = 4 bits
- Default = 0000b
- Encoding when Retimer Target Link Speed is 5.0 GT/s:
 - 0000b -6.0 dB
 - 0001b -3.5 dB
- Encoding when Retimer Target Link Speed is 8.0 GT/s or higher: the Transmitter Preset.

ECN: Base 6.3
Optical $\triangleleft\triangleright$

$\uparrow\downarrow$ When an Optical Retimer Solution is being deployed, the above register fields that are defined per Retimer must have the same value in both Optical Aware Retimers. The setting and maintenance of the values is implementation specific. Each Optical Aware Retimer will have one copy of the registers that are defined per Pseudo Port. \uparrow

A Retimer must examine the values in the above registers when the Retimer exits from Hot Reset. If the Retimer Enter Compliance bit is Set the following rules apply:

- The Retimer adjusts its data rate as defined by Retimer Target Link Speed. No data is forwarded until the data rate change has occurred.
 - $\uparrow\downarrow$ When an Optical Retimer Solution is deployed, the data rate of both the Upstream Pseudo Port and the Downstream Pseudo Port must be adjusted before data is forwarded. \uparrow
- The Retimer configures its Transmitters according to Retimer Compliance Preset/De-emphasis on a per Pseudo Port basis.
- The Retimer must forward the Compliance or Modified Compliance Pattern when it has locked to the pattern. The Retimer must search for the Compliance Pattern if the Retimer Enter Modified Compliance bit is Clear or search for the Modified Compliance Pattern if the Retimer Enter Modified Compliance bit is Set. This occurs independently on each Lane in each direction.
- When using 8b/10b encoding, a particular Lane's Receiver independently determines a successful lock to the incoming Modified Compliance Pattern or Compliance Pattern by looking for any one occurrence of the Modified Compliance Pattern or Compliance Pattern.
 - An occurrence is defined above as the sequence of 8b/10b Symbols defined in § Section 4.2.9.
 - In the case of the Modified Compliance Pattern, the error status Symbols are not to be used for the lock process since they are undefined at any given moment.
 - Lock must be achieved within 1.0 ms of receiving the Modified Compliance Pattern.
- When using 128b/130b or 1b/1b encoding each Lane determines Pattern Lock independently when it achieves Block Alignment as described in § Section 4.2.2.2.1.
 - Lock must be achieved within 1.5 ms of receiving the Modified Compliance Pattern or Compliance Pattern.
- When 128b/130b or 1b/1b encoding is used, Symbols with errors are forwarded unmodified by default, or may optionally be corrected to remove error pollution. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- When 8b/10b encoding is used, Symbols with errors are replaced with the D21.3 Symbol with incorrect disparity by default, or may optionally be corrected to remove error pollution. The default behavior must be supported.
- Once locked, the Retimer keeps an internal count of received Symbol errors, on a per-Lane basis. If the Retimer is forwarding the Modified Compliance Pattern then the error status Symbol when using 8b/10b encoding or the Error_Status field when using 128b/130b encoding is forwarded unmodified by default, or may optionally

be redefined as it is transmitted by the Retimer. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific. The Retimer is permitted to make the pattern lock and Lane error information available in an implementation specific manner, on a per-Lane basis.

- If an EIOS is received on any Lane then:
 - All Lanes in that direction transmit 8 EIOS and then all Transmitters in that direction are placed in Electrical Idle.
 - ↑↑When an Optical Retimer Solution is deployed, the Transmitters placed in Electrical Idle belong to the Pseudo Port in the other Optical Aware Retimer which is connected to the Electrical Link Segment↑
- When both directions have sent 8 EIOS and placed their Transmitters in Electrical Idle the data rate is changed to 2.5 GT/s.
- Set the RT_next_data_rate variable to 2.5 GT/s.
- Set the RT_error_data_rate variable to 2.5 GT/s.
- The Retimer Enter Compliance bit and Retimer Enter Modified Compliance bit are both set to 0b.
- The above additional rules no longer apply unless the rules for entering this section and clause are met again.

 ECN: Base 6.3
 Optical△↔

4.3.8 Execution Mode Rules §

In Execution mode, Retimers directly control all information transmitted by the Pseudo Ports rather than forwarding information.

4.3.8.1 CompLoadBoard Rules §

While the Retimer is in the CompLoadBoard (Compliance Load Board) state both Pseudo Ports are executing the protocol as regular Ports, generating Symbols as specified in the following sub-sections on each Port, rather than forwarding from one Pseudo Port to the other.

Retimers must support the following register field in an implementation specific manner:

- Retimer Compliance SOS
 - One bit per Retimer
 - Type = RWS
 - Size = 1 bit
 - Default = 0b
 - Encoding:
 - 0b = Send no SKP Ordered Sets between sequences when sending the Compliance Pattern or Modified Compliance Pattern with 8b/10b encoding.
 - 1b = Send two SKP Ordered Sets between sequences when sending the Compliance Pattern with 8b/10b encoding.

IMPLEMENTATION NOTE: PASSIVE LOAD ON TRANSMITTER §

This state is entered when a passive load is placed on one Pseudo Port, and the other Pseudo Port is receiving traffic.

4.3.8.1.1 CompLoadBoard.Entry §

- RT_LinkUp = 0b.
- The Pseudo Port that received Compliance Pattern (Pseudo Port A) does the following:
 - The data rate remains at 2.5 GT/s.
 - The Transmitter is placed in Electrical Idle.
 - The Receiver ignores incoming Symbols.
- The other Pseudo Port (Pseudo Port B) does the following:
 - The data rate remains at 2.5 GT/s.
 - The Transmitter is placed in Electrical Idle. Receiver Detection is performed on all Lanes as described in § Section 8.4.5.7.
 - The Receiver ignores incoming Symbols.
- If Pseudo Port B's Receiver Detection determines there are no Receivers attached on any Lanes, then the next state for both Pseudo Ports is CompLoadBoard.Exit.
- Else the next state for both Pseudo Ports is CompLoadBoard.Pattern.
- ↑When an Optical Retimer Solution is deployed, for the remainder of the CompLoadBoard state Pseudo Port B is the Pseudo Port in the other Optical Aware Retimer which is connected to the Electrical Link Segment.↑

ECN: Base 6.3
Optical ▲ ↕

4.3.8.1.2 CompLoadBoard.Pattern §

When The Retimer enters CompLoadBoard.Pattern the following occur:

- Pseudo Port A does the following:
 - The Transmitter remains in Electrical Idle.
 - The Receiver ignores incoming Symbols.
- Pseudo Port B does the following:
 - The Transmitter sends out the Compliance Pattern on all Lanes that detected a Receiver at the data rate and de-emphasis/preset level determined as described in § Section 4.2.7.2.2, (i.e., each consecutive entry into CompLoadBoard advances the pattern), except that the Setting is not set to Setting #1 during Polling.Configuration. Setting #26 and later are not used if Pseudo Port B has received a TS1 or TS2 Ordered Set (or their complement) since the exit of Fundamental Reset. If the new data rate is not 2.5 GT/s, the Transmitter is placed in Electrical Idle prior to the data rate change. The period of Electrical Idle must be greater than 1 ms but it is not to exceed 2 ms.
 - If using 8b/10b encoding and the Retimer Compliance SOS bit (see § Section 4.3.7.15) is Set, send two SKP Ordered Sets between sequences of the Compliance Pattern.

- If Pseudo Port B detects an Electrical Idle exit of any Lane that detected a Receiver, then the next state for both Pseudo Ports is CompLoadBoard.Exit.

4.3.8.1.3 CompLoadBoard.Exit §

When The Retimer enters CompLoadBoard.Exit the following occur:

- The Pseudo Port A:
 - Data rate remains at 2.5 GT/s.
 - The Transmitter sends the Electrical Idle Exit pattern described in § Section 4.3.7.3 , on the Lane(s) where Electrical Idle exit was detected on Pseudo Port B for 1 ms. Then the Transmitter is placed in Electrical Idle.
 - The Receiver ignores incoming Symbols.
- Pseudo Port B:
 - If the Transmitter is transmitting at a rate other than 2.5 GT/s the Transmitter sends eight consecutive EOS .
 - The Transmitter is placed in Electrical Idle. If the Transmitter was transmitting at a rate other than 2.5 GT/s the period of Electrical Idle must be at least 1.0 ms.
 - Data rate is changed to 2.5 GT/s, if not already at 2.5 GT/s.
- Both Pseudo Ports are placed in Forwarding mode.

IMPLEMENTATION NOTE: TS1 ORDERED SETS IN FORWARDING MODE §

Once in Forwarding mode one of two things will likely occur:

- TS1 Ordered Sets are received and forwarded from Pseudo Port's B Receiver to Pseudo Port's A Transmitter. Link training continues.
- Or: TS1 Ordered Sets are not received because 100 MHz pulses are being received on a lane from the compliance load board, advancing the Compliance Pattern. In this case the Retimer must transition from Forwarding mode to CompLoadBoard when the device attached to Pseudo Port A times out from Polling.Active to Polling.Compliance. The Retimer advances the Compliance Pattern on each entry to CompLoadBoard.

4.3.8.2 Link Equalization Rules §

When in the Execution mode performing Link Equalization, the Pseudo Ports act as regular Ports, generating Symbols on each Port rather than forwarding from one Pseudo Port to the other. When the Retimer is in Execution mode it must use the Lane and Link numbers stored in RT_captured_lane_number and RT_captured_link_number .

This mode is entered while the Upstream and Downstream Ports on the Link are in negotiation to enter Phase 2 of the Equalization procedure following the procedure for switching to Execution mode described in § Section 4.3.5 .

↑↑An Optical Retimer Solution must participate in the Execution mode without the need for amended timeout values. Any changes needed to achieve this requirement must be completed in an implementation specific manner prior to the equalization procedure.↑

ECN: Base 6.3
Optical△↔

4.3.8.2.1 Downstream Lanes §

The LF and FS values received in two consecutive TS0 or TS1 Ordered Sets when the Upstream Port is in Phase 1 must be stored for use during Phase 3, if the Downstream Pseudo Port wants to adjust the Upstream Port's Transmitter.

~~When an Optical Retimer Solution is deployed, each Optical Aware Retimer must be aware of the current Equalization phase of the Pseudo Port in the other Optical Aware Retimer, at the other end of the optical extension, which is connected to the Electrical Link Segment (i.e., the Upstream Pseudo Port referred to in this section is in the other Optical Aware Retimer). The phase information is required by the local Pseudo Port to traverse the Equalization phases.~~

 ECN: Base 6.3
 Optical△↔

4.3.8.2.1.1 Phase 1 §

Transmitter behaves as described in § Section 4.2.7.4.2.1.1 except as follows:

- If the data rate of operation is 64.0 GT/s or above, the Retimer Equalization Extend bit of the transmitted TS0 Ordered Sets is set to 1b when the Upstream Pseudo Port state is Phase 0. The EC field is initially set to 00b. After two consecutive TS0 Ordered Sets are received with Retimer Equalization Extend bit set to 0b, the EC field is set to 01b. The Retimer Equalization Extend bit of the transmitted TS0 Ordered Sets it is set to 0b when the Upstream Pseudo Port state is Phase 1. The 24 ms timeout is decreased to 22 ms.

4.3.8.2.1.2 Phase 2 §

Transmitter behaves as described in § Section 4.2.7.4.2.1.2 except as follows:

- If the data rate of operation is 16.0 GT/s or above, the Retimer Equalization Extend bit of the transmitted TS1 Ordered Sets is set to 1b when the Upstream Pseudo Port state is Phase 2 Active, and it is set to 0b when the Upstream Pseudo Port state is Phase 2 Passive.
- Next phase is Phase 3 Active if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=11b.
- Else, if the data rate of operation is less than 64.0 GT/s: next state is Force Timeout after a 32 ms timeout with a tolerance of -0 ms and +4 ms.
- Else, if the data rate of operation is 64.0 GT/s: next state is Force Timeout after a 64 ms timeout with a tolerance of -0 ms and +4 ms.

4.3.8.2.1.3 Phase 3 Active §

If the data rate of operation is 8.0 GT/s then the transmitter behaves as described in § Section 4.2.7.4.2.1.3 except the 24 ms timeout is 2.5 ms and as follows:

- Next phase is Phase 3 Passive if all configured Lanes are operating at their optimal settings.
- Else, next state is Force Timeout after a timeout of 2.5 ms with a tolerance of -0 ms and +0.1 ms

If the data rate of operation is 16.0 GT/s or 32.0 GT/s then the transmitter behaves as described in § Section 4.2.7.4.2.1.3 except the 24 ms timeout is 22 ms and as follows:

- The Retimer Equalization Extend bit of transmitted TS1 Ordered Sets is set to 0b.

- Next phase is Phase 3 Passive if all configured Lanes are operating at their optimal settings and all configured Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
- Else, next state is Force Timeout after a timeout of 22 ms with a tolerance of -0 ms and +1.0 ms.

If the data rate of operation is 64.0 GT/s then the transmitter behaves as described in § Section 4.2.7.4.2.1.3 except the 48 ms timeout is 46 ms and as follows:

- The Retimer Equalization Extend bit of transmitted TS0 or TS1 Ordered Sets is set to 0b.
- Next phase is Phase 3 Passive if all configured Lanes are operating at their optimal settings and all configured Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
- Else, next state is Force Timeout after a timeout of 46 ms with a tolerance of -0 ms and +1.0 ms.

4.3.8.2.1.4 Phase 3 Passive §

- Transmitter sends TS1 Ordered Sets with EC = 11b, Retimer Equalization Extend = 0b, and the Transmitter Preset field and the Coefficients fields must not be changed from the final value transmitted in Phase 3 Active.
- The transmitter switches to Forwarding mode when the Upstream Pseudo Port exits Phase 3.

4.3.8.2.2 Upstream Lanes §

The LF and FS values received in two consecutive TS0 / TS1 Ordered Sets when the Downstream Port is in Phase 1 must be stored for use during Phase 2, if the Upstream Pseudo Port wants to adjust the Downstream Port's Transmitter.

↑↑When an Optical Retimer Solution is deployed, each Optical Aware Retimer must be aware of the current Equalization phase of the Pseudo Port in the other Optical Aware Retimer, at the other end of the optical extension, which is connected to the Electrical Link Segment (i.e., the Downstream Pseudo Port referred to in this section is in the other Optical Aware Retimer). The phase information is required by the local Pseudo Port to traverse the Equalization phases.↑

ECN: Base 6.3
Optical△↔

4.3.8.2.2.1 Phase 0 §

Transmitter follows Phase 0 rules for Upstream Lanes in § Section 4.2.7.4.2.2.1 except as follows:

- If the data rate of operation is 64.0 GT/s or above, the Retimer Equalization Extend bit of the transmitted TS0 Ordered Sets is set to 1b when the Downstream Pseudo Port state is Phase 1 and is transmitting TS0 Ordered Sets with the EC = 00b otherwise it is set to 0b; the 12 ms timeout is 10 ms.

4.3.8.2.2.2 Phase 1 Active §

Transmitter follows Phase 1 rules for Upstream Lanes in § Section 4.2.7.4.2.2.2 .

4.3.8.2.2.3 Phase 2 Active §

If the data rate of operation is 8.0 GT/s then the transmitter behaves as described in § Section 4.2.7.4.2.2.3 except the 24 ms timeout is decreased to 2.5 ms and as follows:

- Next state is Phase 2 Passive if all configured Lanes are operating at their optimal settings.
 • Else, next state is Force Timeout after a 2.5 ms timeout with a tolerance of -0 ms and +0.1 ms

If the data rate of operation is 16.0 GT/s or 32.0 GT/s then the transmitter behaves as described in § [Section 4.2.7.4.2.2.3](#) except the 24 ms timeout is 22 ms and as follows:

- The Retimer Equalization Extend bit of transmitted TS1 Ordered Sets is set to 0b.
- Next phase is Phase 2 Passive if all configured Lanes are operating at their optimal settings and all configured Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
- Else, next state is Force Timeout after a 22 ms timeout with a tolerance of -0 ms and +1.0 ms.

If the data rate of operation is 64.0 GT/s then the transmitter behaves as described in § [Section 4.2.7.4.2.2.3](#) except the 48 ms timeout is 46 ms and as follows:

- The Retimer Equalization Extend bit of transmitted TS0 Ordered Sets is set to 0b.
- Next phase is Phase 2 Passive if all configured Lanes are operating at their optimal settings and all configured Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
- Else, next state is Force Timeout after a 46 ms timeout with a tolerance of -0 ms and +1.0 ms.

4.3.8.2.2.4 Phase 2 Passive §

- Transmitter sends TS0 Ordered Sets with EC = 10b, Retimer Equalization Extend = 0b, and the Transmitter Preset field and the Coefficients fields must not be changed from the final value transmitted in Phase 2 Active.
- If the data rate of operation is 8.0 GT/s, the next state is Phase 3 when the Downstream Pseudo Port has completed Phase 3 Active.
- If the data rate of operation is 16.0 GT/s or above, the next state is Phase 3 when the Downstream Pseudo Port has started Phase 3 Active.

4.3.8.2.2.5 Phase 3 §

Transmitter follows Phase 3 rules for Upstream Lanes in § [Section 4.2.7.4.2.2.4](#) except as follows:

- If the data rate of operation is 16.0 GT/s or above, the Retimer Equalization Extend bit of the transmitted TS1 Ordered Sets is set to 1b when the Downstream Pseudo Port state is Phase 3 Active, and it is set to 0b when the Downstream Pseudo Port state is Phase 3 Passive.
- If all configured Lanes receive two consecutive TS1 Ordered Sets with EC=00b then the Retimer switches to Forwarding mode.
- Else, if the data rate of operation is less than 64.0 GT/s: next state is Force Timeout after a timeout of 32 ms with a tolerance of -0 ms and +4 ms.
- Else, if the data rate of operation is 64.0 GT/s: next state is Force Timeout after a timeout of 64 ms with a tolerance of -0 ms and +4 ms.

4.3.8.2.3 Force Timeout §

- ↑↑When an Optical Retimer Solution is deployed, the Pseudo Ports referred to in this section are the Upstream Pseudo Port in one Optical Aware Retimer and the Downstream Pseudo in the other Optical Aware Retimer.↑
ECN: Base 6.3
Optical△↔
- The Electrical Idle Exit Pattern described in § Section 4.3.7.3 is transmitted by both Pseudo Ports at the current data rate for a minimum of 1.0 ms.
- If on any Lane, a Receiver receives an EIOS or infers Electrical Idle via not detecting an exit from Electrical Idle (see § Table 4-77) then, the Transmitters on all Lanes of the opposite Pseudo Port send an EOSQ and are then placed in Electrical Idle.
- If both Paths have placed their Transmitters in Electrical Idle then, the RT_next_data_rate is set to the RT_error_data_rate , and the RT_error_data_rate is set to 2.5 GT/s, on both Pseudo Ports, and the Retimer enters Forwarding mode.
 - The Transmitters of both Pseudo Ports must be in Electrical Idle for at least 6 μ s, before forwarding data.
- Else after a 96 ms timeout, the RT_next_data_rate is set to 2.5 GT/s and the RT_error_data_rate is set to 2.5 GT/s, on both Pseudo Ports, and the Retimer enters Forwarding mode.

IMPLEMENTATION NOTE: PURPOSE OF FORCE TIMEOUT STATE §

The purpose of this state is to ensure both Link Components are in Recovery Speed at the same time so they go back to the previous data rate.

4.3.8.3 Follower Loopback §

Retimers optionally support Follower Loopback in Execution mode at 8b/10b and 128b/130b encoding. Follower Loopback in Execution mode must be supported at 1b/1b. By default, Retimers are configured to forward loopback between Loopback Lead and Loopback Follower . At 8b/10b and 128b/130b, Retimers are permitted to allow configuration in an implementation specific manner to act as a Loopback Follower on either Pseudo Port; at 1b/1b Loopback Follower on either Pseudo Port is mandatory. The other Pseudo Port that is not the Loopback Follower , places its Transmitter in Electrical Idle, and ignores any data on its Receivers.

↑↑When an Optical Retimer Solution is deployed, the Loopback Follower entry on one Pseudo Port of the Optical Aware Retimer must be communicated to the other Optical Aware Retimer so that it can place the Transmitters of its Pseudo Port, which is connected to the Electrical Link Segment, into Electrical Idle. References to the “other Pseudo Port” in this section are to this remote Pseudo Port.↑
ECN: Base 6.3
Optical△↔

4.3.8.3.1 Follower Loopback.Entry §

The Pseudo Port that is not operating as the Loopback Follower does the following:

- The Transmitter is placed in Electrical Idle.
- The Receiver ignores incoming Symbols.

The Pseudo Port that is operating as the Loopback Follower behaves as the Loopback Follower as described in § Section 4.2.7.10.1 with the following exceptions:

- The statement “LinkUp = 0b (False)” is replaced by “RT_LinkUp = 0b”.
- The statement “If Loopback.Entry was entered from Configuration.Linkwidth.Start” is replaced by “If Follower. Loopback.Entry was entered when RT_LinkUp =0b”.
- References to Loopback.Active become Follower Loopback.Active.

4.3.8.3.2 Follower Loopback.Active §

The Pseudo Port that is not operating as the Loopback Follower does the following:

- The Transmitter remains in Electrical Idle.
- The Receiver continues to ignore incoming Symbols.

The Pseudo Port that is operating the Loopback Follower behaves as the Loopback Follower as described in § Section 4.2.7.10.2 with the following exception:

- References to Loopback.Exit become Follower Loopback.Exit.

4.3.8.3.3 Follower Loopback.Exit §

The Pseudo Port that is not operating as the Loopback Follower must do the following:

- Maintain the Transmitter in Electrical Idle.
- Set the data rate to 2.5 GT/s.
- The Receiver continues to ignore incoming Symbols.

The Pseudo Port that is operating as the Loopback Follower must behave as the Loopback Follower as described in § Section 4.2.7.10.3 with the following exception:

- The clause “The next state of the Loopback Lead and Loopback Follower is Detect” becomes “The Data rate is set to 2.5 GT/s and then both Pseudo Ports are placed in Forwarding mode”.

4.3.9 Retimer Latency §

This Section defines the requirements on allowed Retimer Latency.

4.3.9.1 Measurement §

Latency must be measured when the Retimer is in Forwarding mode and the Link is in L0 , and is defined as the time from when the last bit of a Symbol is received at the input pins of one Pseudo Port to when the equivalent bit is transmitted on the output pins of the other Pseudo Port.

Retimer vendors are strongly encouraged to specify the latency of the Retimer in their data sheets.

Retimers are permitted to have different latencies at different data rates, and when this is the case the latency *MUST@FLIT* be specified per data rate.

4.3.9.2 Maximum Limit on Retimer Latency §

Retimer latency shall be less than the following limit, when not operating in SRIS.

Table 4-78 Retimer Latency Limit not SRIS (Symbol times) §

| | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s |
|-----------------|----------|----------|----------|-----------|-----------|-----------|
| Maximum Latency | 32 | 32 | 64 | 128 | 256 | 512 |

4.3.9.3 Impacts on Upstream and Downstream Ports §

Retimers will add to the channel latency. The round trip delay is 4 times the specified latency when two Retimers are present. It is recommended that designers of Upstream and Downstream Ports consider Retimer latency when determining the following characteristics:

- Data Link Layer Retry Buffer size
- Transaction Layer Receiver buffer size and Flow Control Credits
- Data Link Layer REPLAY_TIMER Limits

Additional buffering (replay or FC) may be required to compensate for the additional channel latency.

4.3.10 SRIS §

Retimers are permitted but not required to support SRIS. Retimers that support SRIS must provide a mechanism for enabling the higher rate of SKP Ordered Set transmission, as Retimers must generate SKP Ordered Sets while in Execution mode. Retimers that are enabled to support SRIS will incur additional latency in the elastic store between receive and transmit clock domains. The additional latency is required to handle the case where a Tx_MPS_Limit TLP is transmitted and SKP Ordered Sets, which are scheduled, are not sent. The additional latency is a function of Link width and Max_Payload_Size. This additional latency is not included in § Table 4-78.

A SRIS capable Retimer must provide an implementation specific mechanism to configure its supported Max_Payload_Size while in SRIS, which must be greater than or equal to the largest Tx_MPS_Limit for the Transmitter in the Port that the Pseudo Port is receiving. Retimer latency must be less than the following limit for the current supported Max_Payload_Size, with SRIS.

Table 4-79 Retimer Latency Limit SRIS (Symbol times) §

| Max_Payload_Size | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s |
|------------------|----------|----------|-----------|-----------|-----------|-----------|
| 128 Bytes | 34 (max) | 34 (max) | 66 (max) | 130 (max) | 258 (max) | 514 (max) |
| 256 Bytes | 36 (max) | 36 (max) | 68 (max) | 132 (max) | 260 (max) | 516 (max) |
| 512 Bytes | 39 (max) | 39 (max) | 71 (max) | 135 (max) | 263 (max) | 519 (max) |
| 1024 Bytes | 46 (max) | 46 (max) | 78 (max) | 142 (max) | 270 (max) | 526 (max) |
| 2048 Bytes | 59 (max) | 59 (max) | 91 (max) | 155 (max) | 283 (max) | 539 (max) |
| 4096 Bytes | 86 (max) | 86 (max) | 118 (max) | 182 (max) | 310 (max) | 566 (max) |

IMPLEMENTATION NOTE: RETIMER LATENCY WITH SRIS CALCULATION: §

§ Table 4-79 is calculated assuming that the link is operating at x1 Link width. The max Latency is the sum of § Table 4-78 and the additional latency required in the elastic store for SRIS clock compensation. The SRIS additional latency in symbol times required for SRIS clock compensation is described in the following equation:

$$2 * \left\lceil \frac{((SRIS\ Link\ Payload\ Size+TLP\ Overhead) / Link\ Width)}{SKP_rate} \right\rceil$$

Equation 4-3 Retimer Latency with SRIS §

Where:

SRIS Link Payload Size

is the value programmed in the Retimer.

TLP Overhead

Represents the additional TLP components which consume Link bandwidth (TLP Prefix, header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.

Link Width

The operating width of the Link.

SKP_rate

The rate that a transmitter schedules SKP Ordered Sets when using 8b/10b encoding, 154, see § Section 4.2.8.4 . When using the 128b/130b encoding the effective rate is the same.

The nominal latency would be ½ of the SRIS additional latency, and is the nominal fill of the elastic store. This makes a worse case assumption that every blocked SKP Ordered Set requires an additional symbol of latency in the elastic store. When an Rx_MPS_Limit size TLP is transmitted, the actual fill of the elastic store could go to zero, or two times the nominal fill depending on the relative clock frequencies. Link width down configure may occur at any time, a lane fails for example, and this down configure may occur faster than the Retimer is able to adjust its nominal elastic store. By default Retimer's will configure its nominal fill based on x1 link width, regardless of the actual current link width.

Retimers that optionally support SRIS , may optionally support a dynamic elastic store. Dynamic elastic store changes the nominal buffer fill as the link width changes. Retimers are permitted to delay the Link LTSSM transitions, only while the Link down configures, in Configuration, for up to 40 µs. Retimers are permitted to delay the TS1 Order Set to TS2 Ordered Set transition between Configuration.Lanenum.Accept and Configuration.Complete to increase their elastic store.

4.3.11 L1 PM Substates Support §

The following Section describes the Retimer's requirements to support the optional L1 PM Substates.

The Retimer enters L1.1 when CLKREQ# is sampled as deasserted. The following occur:

- REFCLK to the Retimer is turned off.
- The PHY remains powered.
- The Retimer places all Transmitters in Electrical Idle on both Pseudo Ports (if not already in Electrical Idle, the expected state). Transmitters maintain their common mode voltage.
- The Retimer must ignore any Electrical Idle exit from all Receivers on both Pseudo Ports.

The Retimer exits L1.1 when CLKREQ# is sampled as asserted. The following occur:

- REFCLK to the Retimer is enabled.
- Normal operation of the Electrical Idle exit circuit is resumed on all Lanes of both Pseudo Ports of the Retimer.
- Normal exit from Electrical Idle exit behavior is resumed, See § Section 4.3.7.3.

Retimers do not support L1.2, but if they support L1.1 and the removal of the reference clock then they must not interfere with the attached components' ability to enter L1.2.

Retimer vendors must document specific implementation requirements applying to CLKREQ#. For example, a Retimer implementation that does not support the removal of the reference clock might require an implementation to pull CLKREQ# low.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: CLKREQ# CONNECTION TOPOLOGY WITH A RETIMER SUPPORTING L1 PM SUBSTATES §

In this platform configuration Downstream Port (A) has only a single CLKREQ# signal. The Upstream and Downstream Ports' CLKREQ# (A and C), and the Retimer's CLKREQB# signals are connected to each other. In this case, Downstream Port (A), must assert CLKREQ# signal whenever it requires a reference clock. Component A, Component B, and the Retimer have their REFCLKs removed/restored at the same time.

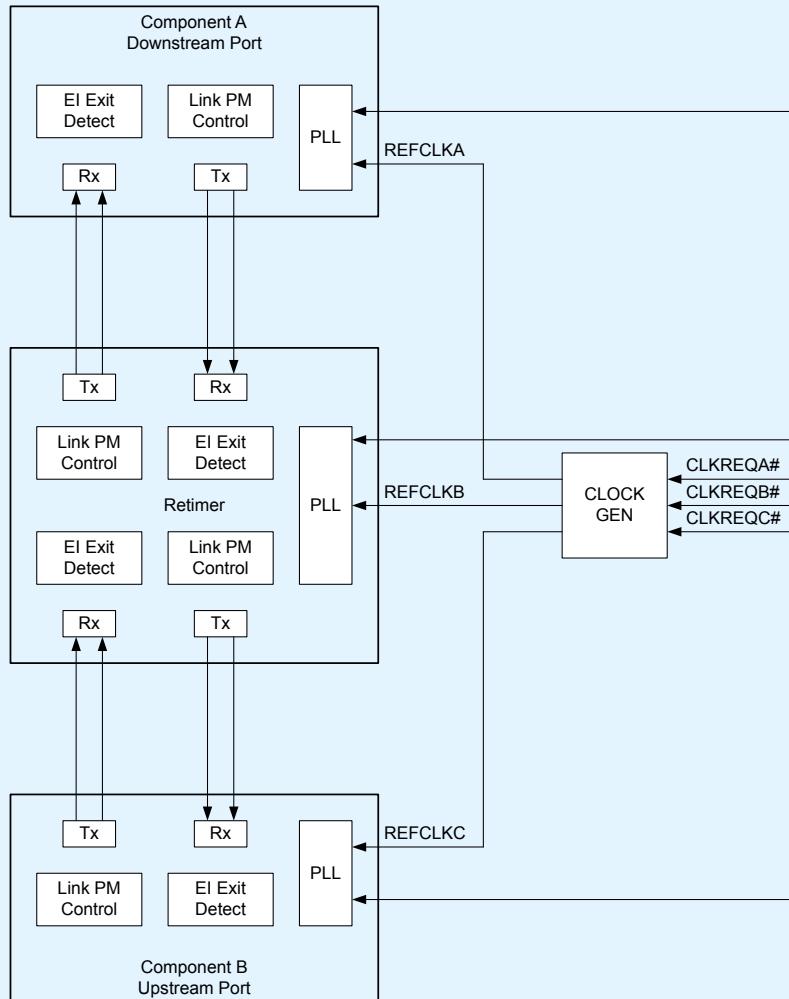


Figure 4-87 Retimer CLKREQ# Connection Topology §

4.3.12 Retimer Configuration Parameters §

Retimers must provide an implementation specific mechanism to configure each of the parameters in this section.

The parameters are split into two groups: parameters that are configurable globally for the Retimer and parameters that are configurable for each physical Retimer Pseudo Port.

If a per Pseudo Port parameter only applies to an Upstream or a Downstream Pseudo Port the Retimer is not required to provide an implementation specific mechanism to configure the parameter for the other type of Pseudo Port.

↑↑When an Optical Retimer Solution is deployed, the global parameters apply to the Optical Aware Retimers at both ends of the optical extension. Each Optical Aware Retimer has its own set of Per Physical Pseudo Port Parameters.↑

 ECN: Base 6.3
 Optical

4.3.12.1 Global Parameters §

- **Port Orientation Method** . This controls whether the Port Orientation is determined dynamically as described in § Section 4.3.7.2 , or statically based on vendor assignment of Upstream and Downstream Pseudo Ports. If the Port Orientation is set to static the Retimer is not required to dynamically adjust the Port Orientation as described in § Section 4.3.7.2 . The default behavior is for the Port Orientation to be dynamically determined.
- **Maximum Data Rate** . This controls the maximum data rate that the Retimer sets in the Data Rate Identifier field of training sets that the Retimer transmits. Retimers that support only the 2.5 GT/s speed are permitted not to provide this configuration parameter.
- **SRIS Enable** . This controls whether the Retimer is configured for SRIS and transmits SKP Ordered Sets at the SRIS mode rate when in Execution mode. Retimers that do not support SRIS and at least one other clocking architecture are not required to provide this configuration parameter.
- **SRIS Link Payload Size** . This controls the maximum payload size the Retimer supports while in SRIS. The value must be selectable from all the Maximum Payload Sizes shown in § Table 4-79 . The default value of this parameter is to support a maximum payload size of 4096 bytes. Retimers that do not support SRIS are not required to provide this configuration parameter.

The following are examples of cases where it might be appropriate to configure the SRIS Link Payload Size to a smaller value than the default:

- A Retimer is part of a motherboard connected to Root Port that supports a maximum payload size less than 4096 bytes.
- A Retimer is part of an adapter connected to an Upstream Port (e.g., Endpoint) that supports a maximum payload size less than 4096 bytes.
- A Retimer is located Downstream of the Downstream Port of a Switch integrated as part of a system, the Root Port supports a maximum payload size less than 4096 bytes and the system does not support peer to peer traffic.
- **Enhanced Link Behavior Control** . This controls the ability for the Retimer to either bypass equalization to the highest data rate or completely bypass equalization when it supports 32.0 GT/s.

4.3.12.2 Per Physical Pseudo Port Parameters §

- **Port Orientation** . This is applicable only when the Port Orientation Method is configured for static determination. This is set for either Upstream or Downstream. Each Pseudo Port must be configured for a different orientation, or the behavior is undefined.
- **Selectable De-emphasis** . When the Downstream Pseudo Port is operating at 5.0 GT/s this controls the transmit de-emphasis of the Link to either -3.5 dB or -6 dB in specific situations and the value of the Selectable De-emphasis field in training sets transmitted by the Downstream Pseudo Port. See § Section 4.2.7 for detailed

usage information. When the Link Segment is not operating at the 5.0 GT/s speed, the setting of this bit has no effect. Retimers that support only the 2.5 GT/s speed are permitted not to provide this configuration parameter.

- **Rx Impedance Control** . This controls whether the Retimer dynamically applies and removes $50\ \Omega$ terminations or statically has $50\ \Omega$ terminations present. The value must be selectable from Dynamic, Off, and On. The default behavior is Dynamic.
- **Tx Compliance Disable** . This controls whether the Retimer transmits the Compliance Pattern in the CompLoadBoard.Pattern state. The default behavior is for the Retimer to transmit the Compliance Pattern in the CompLoadBoard.Pattern state. If TX Compliance Pattern is set to disabled, the Retimer Transmitters remain in Electrical Idle and do not transmit Compliance Pattern in CompLoadBoard.Pattern - all other behavior in the CompLoadBoard state is the same.
- **Pseudo Port Follower Loopback** . This controls whether the Retimer operates in a Forwarding mode during loopback on the Link or enters Follower Loopback on the Pseudo Port. The default behavior is for the Retimer to operate in Forwarding mode during loopback. Retimers that do not support optional Follower Loopback are permitted not to provide this configuration parameter. This configuration parameter shall only be enabled for one physical Port. Retimer behavior is undefined if the parameter is enabled for more than one physical Port.
- **Downstream Pseudo Port 8GT TX Preset** . This controls the initial TX preset used by the Downstream Pseudo Port transmitter for 8.0 GT/s transmission. The default value is implementation specific. The value must be selectable from all applicable values in § Table 4-34 .
- **Downstream Pseudo Port 16GT TX Preset** . This controls the initial TX preset used by the Downstream Pseudo Port transmitter for 16.0 GT/s transmission. The default value is implementation specific. The value must be selectable from all applicable values in § Table 4-34 .
- **Downstream Pseudo Port 32GT TX Preset** . This controls the initial TX preset used by the Downstream Pseudo Port transmitter for 32.0 GT/s transmission. The default value is implementation specific. The value must be selectable for all applicable values in § Table 4-34 .
- **Downstream Pseudo Port 64GT TX Preset** . This controls the initial TX preset used by the Downstream Pseudo Port transmitter for 64.0 GT/s transmission. The default value is implementation specific. The value must be selectable for all applicable values in § Table 4-34 .
- **Downstream Pseudo Port 8GT Requested TX Preset** . This controls the initial transmitter preset value used in the EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 8.0 GT/s. The default value is implementation specific. The value must be selectable from all values in § Table 4-34 .
- **Downstream Pseudo Port 16GT Requested TX Preset** . This controls the initial transmitter preset value used in the 128b/130b EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 16.0 GT/s. The default value is implementation specific. The value must be selectable from all values in § Table 4-34 .
- **Downstream Pseudo Port 32GT Requested TX Preset** . This controls the initial transmitter preset value used in the 128b/130b EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 32.0 GT/s. The default value is implementation specific. The value must be selectable from all values in § Table 4-34 .
- **Downstream Pseudo Port 64GT Requested TX Preset** . This controls the initial transmitter preset value used in the 128b/130b EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 64.0 GT/s. The default value is implementation specific. The value must be selectable from all values in § Table 4-34 .
- **Downstream Pseudo Port 8GT RX Hint** . This controls the Receiver Preset Hint value used in the EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 8.0 GT/s. The default value is implementation specific. The value must be selectable from all values in § Table 4-35 .

4.3.13 In Band Register Access §

- Retimers operating at 16.0 GT/s or higher may optionally support inband read only access. Control SKP Ordered Sets at 16.0 GT/s or higher provide the mechanism via the Margin Command ‘Access Retimer Register’,

see § Table 4-75 . Retimers that support inband read only access must return a non-zero value for the DWORD at Registers offsets 80h and 84h. Retimers that do not support inband read only access must return a zero value.

- Register offsets between A0h and FFh are designated as Vendor Defined register space.
- Register offsets from 00h to 7Fh and 85H to 9Fh are Reserved for PCI-SIG future use.

Base 6.4 vs Base 6.3

5. Power Management §

This chapter describes power management (PM) capabilities and protocols.

5.1 Overview §

Power Management states are as follows:

- D states are associated with a particular Function
 - D0 is the operational state and consumes the most power
 - D1 and D2 are intermediate power saving states
 - D3_{Hot} is a very low power state
 - D3_{Cold} is the power off state
- L states are associated with a particular Link
 - L0 is the operational state
 - L0p is a reduced power sub-state of L0 (see § Section 4.2.6.7)
 - L0s , L1 , L1.0 , L1.1 , and L1.2 are various lower power states

Other specifications define related power states (e.g., S states). This specification does not describe relationships between those states and D/L states.

PM provides the following services:

- A mechanism to identify power management capabilities of a given Function
- The ability to transition a Function into a certain power management state
- Notification of the current power management state of a Function
- The option to wakeup the system on a specific event

PM is compatible with the *PCI Bus Power Management Interface Specification* and the *Advanced Configuration and Power Interface Specification*. This chapter also defines PCI Express Native Power Management extensions.

PM defines Link power management states that a PCI Express physical Link is permitted to enter in response to either software driven D-state transitions or active state Link power management activities. PCI Express Link states are not visible directly to legacy bus driver software, but are derived from the power management state of the components residing on those Links. Defined Link states are L0 , L0s , L1 , L2 , and L3 . The power savings increase as the Link state transitions from L0 through L3 .

Components may wakeup the system using a wakeup mechanism followed by a power management event (PME) Message. PCI Express systems may provide the optional auxiliary power supply (Vaux) needed for wakeup operation from states where the main power supplies are off.

The specific definition and requirements associated with **Vaux** are form-factor specific, and throughout this document the terms “auxiliary power” and “Vaux” should be understood in reference to the specific form factor in use.

Unlike earlier mechanisms, the PCI Express-PM PME mechanism separates the following two PME tasks:

- Reactivation (wakeup) of the associated resources (i.e., re-establishing reference clocks and main power rails to the PCI Express components)
- Sending a PME Message to the Root Complex to provide the of the wakeup event

Active State Power Management (ASPM) is an autonomous hardware-based, active state mechanism that enables power savings even when the connected components are in the D0 state. After a period of idle Link time, an ASPM Physical-Layer protocol places the idle Link into a lower power state. Once in the lower-power state, transitions to the fully operative L0 state are triggered by traffic appearing on either side of the Link. ASPM may be disabled by software. Refer to § [Section 5.4.1](#) for more information on ASPM.

5.2 Link State Power Management §

PCI Express defines Link power management states, replacing the bus power management states that were defined by the *PCI Bus Power Management Interface Specification*. Link states are not visible to PCI-PM legacy compatible software, and are either derived from the power management D-states of the corresponding components connected to that Link or by ASPM protocols (see § [Section 5.4.1](#)).

Note that the PCI Express Physical Layer may define additional intermediate states. Refer to § [Chapter 4](#) for more detail on each state and how the Physical Layer handles transitions between states.

PCI Express-PM defines the following Link power management states:

- L0 (including the L0p sub-state) - Active state.
L0 support is required for both ASPM and PCI-PM compatible power management.
All PCI Express transactions and other operations are enabled.
- L0s - A low resume latency, energy saving “standby” state.
L0s support is optional for ASPM unless the applicable form factor specification for the Link explicitly requires L0s support.
All main power supplies, component reference clocks, and components’ internal PLLs must be active at all times during L0s. TLP and DLLP transmission is disabled for a Port whose Link is in Tx_L0s.

The Physical Layer provides mechanisms for quick transitions from this state to the L0 state. When common (distributed) reference clocks are used on both sides of a Link, the transition time from L0s to L0 is desired to be less than 100 Symbol Times.

It is possible for the Transmit side of one component on a Link to be in L0s while the Transmit side of the other component on the Link is in L0.

- L1 - Higher latency, lower power “standby” state.
L1 support is required for PCI-PM compatible power management. L1 is optional for ASPM unless specifically required by a particular form factor.

When L1 PM Substates is enabled by setting one or more of the enable bits in the L1 PM Substates Control 1 Register this state is referred to as the L1.0 substate.

All main power supplies must remain active during L1. As long as they adhere to the advertised L1 exit latencies, implementations are explicitly permitted to reduce power by applying techniques such as, but not limited to, periodic rather than continuous checking for Electrical Idle exit, checking for Electrical Idle exit on only one Lane, and powering off of unneeded circuits. All platform-provided component reference clocks must remain active during L1, except as permitted by Clock Power Management (using CLKREQ#) and/or L1 PM

Substates when enabled. A component's internal PLLs may be shut off during L1, enabling greater power savings at a cost of increased exit latency.¹⁰³

The L1 state is entered whenever all Functions of a Downstream component on a given Link are programmed to a D-state other than D0. The L1 state also is entered if the Downstream component requests L1 entry (ASPM) and receives positive acknowledgement for the request.

Exit from L1 is initiated by an Upstream-initiated transaction targeting a Downstream component, or by the Downstream component's initiation of a transaction heading Upstream. Transition from L1 to L0 is desired to be a few microseconds.

TLP and DLLP transmission is disabled for a Link in L1.

- **L1 PM Substates** - optional L1.1 and L1.2 substates of the L1 low power Link state for PCI-PM and ASPM.

In the L1.1 substate, the Link common mode voltages are maintained. The L1.1 substate is entered when the Link is in the L1.0 substate and conditions for entry into L1.1 substate are met. See § Section 5.5.1 for details.

In the L1.2 substate, the Link common mode voltages are not required to be maintained. The L1.2 substate is entered when the Link is in the L1.0 substate and conditions for entry into L1.2 substate are met. See § Section 5.5.1 for details.

Exit from all L1 PM Substates is initiated when the CLKREQ# signal is asserted (see § Section 5.5.2.1 and § Section 5.5.3.3).

- **L2/L3 Ready** - Staging point for L2 or L3.

L2/L3 Ready transition protocol support is required.

L2/L3 Ready is a pseudo-state (corresponding to the LTSSM L2 state) that a given Link enters when preparing for the removal of power and clocks from the Downstream component or from both attached components. This process is initiated after PM software transitions a device into a D3 state, and subsequently calls power management software to initiate the removal of power and clocks. After the Link enters the L2/L3 Ready state the component(s) are ready for power removal. After main power has been removed, the Link will either transition to L2 if Vaux is provided and used, or it will transition to L3 if no Vaux is provided or used. Note that these are PM pseudo-states for the Link; under these conditions, the LTSSM will in, general, operate only on main power, and so will power off with main power removal.

The L2/L3 Ready state entry transition process must begin as soon as possible following the acknowledgment of a PME_Turn_Off Message, (i.e., the injection of a PME_TO_Ack TLP). The Downstream component initiates L2/L3 Ready entry by sending a PM_Enter_L23 DLLP. Refer to § Section 5.7 for further detail on power management system Messages.

TLP and DLLP transmission is disabled for a Link in L2/L3 Ready.

Note: Exit from L2/L3 Ready back to L0 will be through intermediate LTSSM states. Refer to § Chapter 4. for detailed information.

- L2 - Auxiliary-powered Link, deep-energy-saving state.

L2 support is optional, and dependent upon the presence of auxiliary power.

A component may only consume auxiliary power if enabled to do so as described in § Section 5.6.

In L2, the component's main power supply inputs and reference clock inputs are shut off.

When in L2, any Link reactivation wakeup logic (Beacon or WAKE#), PME context, and any other "keep alive" logic is powered by auxiliary power.

¹⁰³. For example, disabling the internal PLL may be something that is desirable when in D3_{Hot}, but not so when in D1 or D2.

TLP and DLLP transmission is disabled for a Link in L2 .

- **L3** - Link Off state.

When no power is present, the component is in the L3 state.

- **LDn** - A transitional Link Down pseudo-state prior to L0 .

This pseudo-state is associated with the LTSSM states Detect , Polling , and Configuration , and, when applicable, Disabled , Loopback , and Hot Reset .

Refer to § Section 4.2 for further detail relating to entering and exiting each of the L-states between L0 and L2/L3 Ready (L2.Idle from the § Chapter 4. perspective). The L2 state is an abstraction for PM purposes distinguished by the presence of auxiliary power, and should not be construed to imply a requirement that the LTSSM remain active.

The electrical section specifies the electrical properties of drivers and Receivers when no power is applied. This is the L3 state but the electrical section does not refer to L3 .

§ Figure 5-1 shows an overview of L-state transitions that may occur.

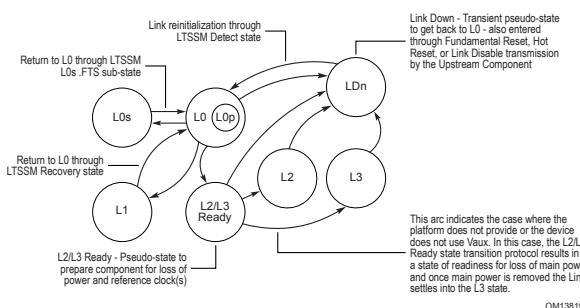


Figure 5-1 Link Power Management State Flow Diagram §

The L1 and L2/L3 Ready entry negotiations happen while in the L0 state. L1 and L2/L3 Ready are entered only after the negotiation completes. Link Power Management remains in L0 until the negotiation process is completed, unless LDn occurs. Note that these states and state transitions do not correspond directly to the actions of the Physical Layer LTSSM. For example in § Figure 5-1 , L0 encompasses the LTSSM L0 , Recovery , and, during LinkUp, Configuration states. Also, the LTSSM is typically powered by main power (not Vaux), so LTSSM will not be powered in either the L2 or the L3 state.

The following example sequence illustrates the multi-step Link state transition process leading up to entering a system sleep state:

1. System software directs all Functions of a Downstream component to D3Hot .
2. The Downstream component then initiates the transition of the Link to L1 as required.
3. System software then causes the Root Complex to broadcast the PME_Turn_Off Message in preparation for removing the main power source.
4. This Message causes the subject Link to transition back to L0 in order to send it and to enable the Downstream component to respond with PME_TO_Ack .
5. After sending the PME_TO_Ack , the Downstream component initiates the L2/L3 Ready transition protocol.

L0 → L1 → L0 → L2/L3 Ready

As the following example illustrates, it is also possible to remove power without first placing all Functions into D3Hot :

1. System software causes the Root Complex to broadcast the PME_Turn_Off Message in preparation for removing the main power source.
2. The Downstream components respond with PME_TO_Ack.
3. After sending the PME_TO_Ack, the Downstream component initiates the L2/L3 Ready transition protocol.

L0 → L2/L3 Ready

The L1 entry negotiation (whether invoked via PCI-PM or ASPM mechanisms) and the L2/L3 Ready entry negotiation map to a state machine which corresponds to the actions described later in this chapter. This state machine is reset to an idle state. For a Downstream component, the first action taken by the state machine, after leaving the idle state, is to start sending the appropriate entry DLLPs depending on the type of negotiation. If the negotiation is interrupted, for example by a trip through Recovery, the state machine in both components is reset back to the idle state. The Upstream component must always go to the idle state, and wait to receive entry DLLPs. The Downstream component must always go to the idle state and must always proceed to sending entry DLLPs to restart the negotiation.

§ Table 5-1 summarizes each L-state, describing when they are used, and the platform and component behaviors that correspond to each.

A “Yes” entry indicates that support is required (unless otherwise noted). “On” and “Off” entries indicate the required clocking and power delivery. “On/Off” indicates an optional design choice.

Table 5-1 Summary of PCI Express Link Power Management States §

| | L-State Description | Used by S/W Directed PM | Used by ASPM | Platform Reference Clocks | Platform Main Power | Component Internal PLL | Platform Vaux |
|-----------------------------------|--|---------------------------|-------------------------------------|---------------------------|---------------------|------------------------|-----------------|
| <u>L0 / L0p</u> | Fully active Link | Yes (<u>D0</u>) | Yes (<u>D0</u>) | On | On | On | On/Off |
| <u>L0s</u> | Standby state | No | Yes ¹ (opt., <u>D0</u>) | On | On | On | On/Off |
| <u>L1</u> | Lower power standby | Yes (<u>D1 - D3Hot</u>) | Yes (opt., <u>D0</u>) | On/Off ⁶ | On | On/Off ² | On/Off |
| <u>L2/L3 Ready</u> (pseudo-state) | Staging point for power removal | Yes ³ | No | On/Off ⁶ | On | On/Off | On/Off |
| <u>L2</u> | Low power sleep state (all clocks, main power off) | Yes ⁴ | No | Off | Off | Off | On ⁵ |
| <u>L3</u> | Off (zero power) | n/a | n/a | Off | Off | Off | Off |
| <u>LDn</u> (pseudo-state) | Transitional state preceding <u>L0</u> | Yes | N/A | On | On | On/Off | On/Off |

Notes:

1. L0s exit latency will be greatest in Link configurations with independent reference clock inputs for components connected to opposite ends of a given Link (vs. a common, distributed reference clock).
2. L1 exit latency will be greatest for components that internally shut off their PLLs during this state.
3. L2/L3 Ready entry sequence is initiated at the completion of the PME_Turn_Off / PME_TO_Ack protocol handshake. It is not directly affiliated with either a D-State transition or a transition in accordance with ASPM policies and procedures.
4. Depending upon the platform implementation, the system's sleep state may use the L2 state, transition to fully off (L3), or it may leave Links in the L2/L3 Ready state. L2/L3 Ready state transition protocol is initiated by the Downstream component following reception and TLP acknowledgement of the PME_Turn_Off TLP Message. While platform support for

| | L-State Description | Used by S/W Directed PM | Used by ASPM | Platform Reference Clocks | Platform Main Power | Component Internal PLL | Platform Vaux |
|--|---------------------|-------------------------|--------------|---------------------------|---------------------|------------------------|---------------|
|--|---------------------|-------------------------|--------------|---------------------------|---------------------|------------------------|---------------|

an L2 sleep state configuration is optional (depending on the availability of Vaux), component protocol support for transitioning the Link to the L2/L3 Ready state is required.

5. L2 is distinguished from the L3 state only by the presence and use of Vaux . After the completion of the L2/L3 Ready state transition protocol and before main power has been removed, the Link has indicated its readiness for main power removal.
6. Low-power mobile or handheld devices may reduce power by clock gating the reference clock(s) via the “clock request” (CLKREQ#) mechanism. As a result, components targeting these devices should be tolerant of the additional delays required to re-energize the reference clock during the low-power state exit.

5.3 PCI-PM Software Compatible Mechanisms §

5.3.1 Device Power Management States (D-States) of a Function §

While the concept of these power states is universal for all Functions in the system, the meaning, or intended functional behavior when transitioned to a given power management state, is dependent upon the type (or class) of the Function.

The D0 power management state is the normal operation state of the Function. Other states are various levels of reduced power, where the Function is either not operating or supports a limited set of operations. D1 and D2 are intermediate states that are intended to afford the system designer more flexibility in balancing power savings, restore time, and low power feature availability tradeoffs for a given device class. The D1 state could, for example, be supported as a slightly more power consuming state than D2 , however one that yields a quicker restore time than could be realized from D2 .

The D3 power management state constitutes a special category of power management state in that a Function could be transitioned into D3 either by software or by physically removing its power. In that sense, the two D3 variants have been designated as **D3_{Hot}** and **D3_{Cold}** where the subscript refers to the presence or absence of main power respectively.

Functions in D3_{Hot} are permitted to be transitioned to the D0 state via software by writing to the Function's PMCSR register. Functions in the D3_{Cold} state are permitted to be transitioned to the D0_{uninitialized} state by reapplying main power and asserting Fundamental Reset.

All Functions must support the D0 and D3 states (both D3_{Hot} and D3_{Cold}). The D1 and D2 states are optional.

When a Type 1 Function associated with a Switch/Root Port (a “virtual bridge”) is in a non-D0 power state, ~~it will emulate~~ the ~~behavior~~ following rules apply¹⁰⁴ :

Errata: Bade 6.3
B833△▷

- All Memory and I/O requests flowing Downstream are terminated as Unsupported Requests.
- All ~~Type 1~~ Configuration Requests are terminated as Unsupported Requests, however ~~Type 0~~ Configuration Request handling is unaffected by the virtual bridge ~~D state.~~ All Memory and I/O requests flowing Downstream are terminated as Unsupported
- Completions flowing in either direction across the virtual bridge are unaffected by the virtual bridge D state.

Errata: Bade 6.3
B833△◁

Note that the handling of Messages is not affected by the PM state of the virtual bridge.

104. ~~In previous versions of this specification, these requirements were incorporated by reference to another specification that is no longer maintained; as a result, the requirements are now stated in its handling of Memory, I/O, and Configuration Requests and Completions.~~ this specification.

5.3.1.1 D0 State §

All Functions must support the D0 state. D0 is divided into two distinct substates, the “un-initialized” substate and the “active” substate. When a component comes out of Conventional Reset all Functions of the component enter the **D0 uninitialized** state. When a Function completes FLR, it enters the **D0 uninitialized** state. After configuration is complete a Function enters the **D0 active** state, the fully operational state for a PCI Express Function. A Function enters the **D0 active** state whenever any single or combination of the Function's Memory Space Enable, I/O Space Enable, or Bus Master Enable bits have been Set¹⁰⁵.

5.3.1.2 D1 State §

D1 support is optional. While in the D1 state, a Function must not initiate any Request TLPs on the Link with the exception of Messages as defined in § Section 2.2.8. Configuration and Message Requests are the only TLPs accepted by a Function in the D1 state. All other received Requests must be handled as Unsupported Requests, and all received Completions may optionally be handled as Unexpected Completions. If an error caused by a received TLP (e.g., an Unsupported Request) is detected while in D1, and reporting is enabled, the Link must be returned to L0 if it is not already in L0 and an error Message must be sent. If an error caused by an event other than a received TLP (e.g., a Completion Timeout) is detected while in D1, an error Message must be sent when the Function is programmed back to the D0 state.

Note that a Function's software driver participates in the process of transitioning the Function from D0 to D1. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the Function for the transition to D1. As part of this quiescence process the Function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D1.

5.3.1.3 D2 State §

D2 support is optional. When a Function is not currently being used and probably will not be used for some time, it may be put into D2. This state requires the Function to provide significant power savings while still retaining the ability to fully recover to its previous condition. While in the D2 state, a Function must not initiate any Request TLPs on the Link with the exception of Messages as defined in § Section 2.2.8. Configuration and Message requests are the only TLPs accepted by a Function in the D2 state. All other received Requests must be handled as Unsupported Requests, and all received Completions may optionally be handled as Unexpected Completions. If an error caused by a received TLP (e.g., an Unsupported Request) is detected while in D2, and reporting is enabled, the Link must be returned to L0 if it is not already in L0 and an error Message must be sent. If an error caused by an event other than a received TLP (e.g., a Completion Timeout) is detected while in D2, an error Message must be sent when the Function is programmed back to the D0 state.

Note that a Function's software driver participates in the process of transitioning the Function from D0 to D2. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the Function for the transition to D2. As part of this quiescence process the Function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D2.

System software must restore the Function to the **D0 active** state before memory or I/O space can be accessed. Initiated actions such as bus mastering and interrupt request generation can only commence after the Function has been restored to **D0 active**.

¹⁰⁵. A Function remains in **D0 active** even if these enable bits are subsequently cleared.

There is a minimum recovery time requirement of 200 µs between when a Function is programmed from D2 to D0 and the next Request issued to the Function. Behavior is undefined for Requests received in this recovery time window (see § Section 7.9.16).

5.3.1.4 D3 State §

D3 support is required, (both the D3Cold and the D3Hot states).

Functional context is required to be maintained by Functions in the D3Hot state if the No_Soft_Reset field in the PMCSR is Set. In this case, System Software is not required to re-initialize the Function after a transition from D3Hot to D0 (the Function will be in the D0active state). If the No_Soft_Reset bit is Clear, functional context is not required to be maintained by the Function in the D3Hot state, however it is not guaranteed that functional context will be cleared and software must not depend on such behavior. As a result, in this case System Software is required to fully re-initialize the Function after a transition to D0 as the Function will be in the D0uninitialized state.

The Function will be reset if the Link state has transitioned to the L2/L3 Ready state regardless of the value of the No_Soft_Reset bit.

IMPLEMENTATION NOTE: TRANSITIONING TO L2/L3 READY §

As described in § Section 5.2 , transition to the L2/L3 Ready state is initiated by platform power management software in order to begin the process of removing main power and clocks from the device. As a result, it is expected that a device will transition to D3Cold shortly after its Link transitions to L2/L3 Ready , making the No_Soft_Reset bit, which only applies to D3Hot , irrelevant. While there is no guarantee of this correlation between L2/L3 Ready and D3Cold , system software should ensure that the L2/L3 Ready state is entered only when the intent is to remove device main power. Device Functions, including those that are otherwise capable of maintaining functional context while in D3Hot (i.e., set the No_Soft_Reset bit), are required to re-initialize internal state as described in § Section 2.9.1 when exiting L2/L3 Ready due to the required DL_Down status indication.

Unless the Immediate_Readiness_on_Return_to_D0 bit in the PCI-PM Power Management Capabilities register is Set, System Software must allow a minimum recovery time following a D3Hot → D0 transition of at least 10 ms (see § Section 7.9.16), prior to accessing the Function. This recovery time may, for example, be used by the D3Hot → D0 transitioning component to bootstrap any of its component interfaces (e.g., from serial ROM) prior to being accessible. Attempts to target the Function during the recovery time (including configuration request packets) will result in undefined behavior.

5.3.1.4.1 D3Hot State §

Configuration and Message requests are the only TLPs accepted by a Function in the D3Hot state. All other received Requests must be handled as Unsupported Requests , and all received Completions may optionally be handled as Unexpected Completions. If an error caused by a received TLP (e.g., an Unsupported Request) is detected while in D3Hot , and reporting is enabled, the Link must be returned to L0 if it is not already in L0 and an error Message must be sent. If an error caused by an event other than a received TLP (e.g., a Completion Timeout) is detected while in D3Hot , an error Message may optionally be sent when the Function is programmed back to the D0 state. Once in D3Hot the Function can later be transitioned into D3Cold (by removing power from its host component).

Note that a Function's software driver participates in the process of transitioning the Function from D0 to D3_{Hot}. It contributes to the process by saving any functional state that would otherwise be lost with removal of main power, and otherwise preparing the Function for the transition to D3_{Hot}. As part of this quiescence process the Function's software driver must ensure that any outstanding transactions (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D3_{Hot}.

Note that D3_{Hot} is also a useful state for reducing power consumption by idle components in an otherwise running system.

Functions that are in D3_{Hot} are permitted to be transitioned by software (writing to their PMCSR PowerState field) to the D0_{active} state or the D0_{uninitialized} state. Functions that are in D3_{Hot} must respond to Configuration Space accesses as long as power and clock are supplied so that they can be returned to D0 by software. Note that the Function is not required to generate an internal hardware reset during or immediately following its transition from D3_{Hot} to D0 (see usage of the No_Soft_Reset bit in the PMCSR).

If not requiring an internal reset, upon completion of the D3_{Hot} to D0 active state, no additional operating system intervention is required beyond writing the PowerState field. If the internal reset is required, devices return to D0_{uninitialized} and a full reinitialization is required on the device. The full reinitialization sequence returns the device to D0 active .

If the device supports PME events, and PME_En is Set, PME context must be preserved in D3_{Hot}. PME context must also be preserved in a PowerState command transition back to D0 .

IMPLEMENTATION NOTE: DEVICES NOT PERFORMING AN INTERNAL RESET §

Bus controllers to non-PCIe buses and resume from D3_{Hot} bus controllers on PCIe buses that serve as interfaces to non-PCIe buses, (e.g., CardBus, USB, and IEEE 1394) are examples of bus controllers that would benefit from not requiring an internal reset upon resume from D3_{Hot}. If this internal reset is not required, the bus controller would not need to perform a downstream bus reset upon resume from D3_{Hot} on its secondary (non-PCIe) bus.

IMPLEMENTATION NOTE: MULTI-FUNCTION DEVICE ISSUES WITH SOFT RESET §

With Multi-Function Devices (MFDs), certain control settings affecting overall device behavior are determined either by the collective settings in all Functions or strictly off the settings in Function 0. Here are some key examples:

- With non-ARI MFDs, certain controls in the Device Control register and Link Control registers operate off the collective settings of all Functions (see § [Section 7.5.3.4](#) and § [Section 7.5.3.7](#)).
- With ARI Devices, certain controls in the Device Control register and Link Control registers operate strictly off the settings in Function 0 (see § [Section 7.5.3.4](#) and § [Section 7.5.3.7](#)).
- With all MFDs, certain controls in the Device Control 2 and Link Control 2 registers operate strictly off the settings in Function 0 (see § [Section 7.5.3.16](#) and § [Section 7.5.3.19](#)).

Performing a soft reset on any Function (especially Function 0) may disrupt the proper operation of other active Functions in the MFD. Since some Operating Systems transition a given Function between D_{3Hot} and D₀ with the expectation that other Functions will not be impacted, it is strongly recommended that every Function in an MFD be implemented with the No_Soft_Reset bit Set in the Power Management Control/Status register. This way, transitioning a given Function from D_{3Hot} to D₀ will not disrupt the proper operation of other active Functions. For Functions that support Flit Mode, the No_Soft_Reset bit is required to be Set (see § [Table 7-15](#)).

It is also strongly recommended that every Endpoint Function in an MFD implement Function Level Reset (FLR) (i.e., Function Level Reset Capability is Set). FLR can be used to reset an individual Endpoint Function without impacting the settings that might affect other Functions, particularly if those Functions are active. As a result of FLR's quiescing, error recovery, and cleansing for reuse properties, FLR is also recommended for single-Function Endpoint devices.

[5.3.1.4.2 D_{3Cold} State](#) §

A Function transitions to the D_{3Cold} state when its main power is removed. A power-on sequence with its associated Cold Reset transitions a Function from the D_{3Cold} state to the D₀ uninitialized state, and the power-on defaults will be restored to the Function by hardware just as at initial power up. At this point, software must perform a full initialization of the Function in order to re-establish all functional context, completing the restoration of the Function to its D_{0active} state.

When PME_En is Set, Functions that support wakeup functionality from D_{3Cold} must maintain their PME context in the PMCSR for inspection by PME service routine software during the course of the resume process. Retention of additional context is implementation specific.

IMPLEMENTATION NOTE: PME CONTEXT §

Examples of PME context include, but are not limited to, a Function's PME_Status bit, the requesting agent's Requester ID, Caller ID if supported by a modem, IP information for IP directed network packets that trigger a resume event, etc.

A Function's PME assertion is acknowledged when system software performs a “write 1 to clear” configuration transaction to the asserting Function's PME_Status bit of its PCI-PM compatible PMCSR.

An auxiliary power source must be used to support PME event detection within a Function, Link reactivation, and to preserve PME context from within D3Cold. Note that once the I/O Hierarchy has been brought back to a fully communicating state, as a result of the Link reactivation, the waking agent then propagates a PME Message to the root of the Hierarchy indicating the source of the PME event. Refer to § Section 5.3.3 for further PME specific detail.

5.3.2 PM Software Control of the Link Power Management State §

The power management state of a Link is determined by the D-state of its Downstream component.

§ Table 5-2 depicts the relationships between the power state of a component (with an Upstream Port) and its Upstream Link.

Table 5-2 Relation Between Power Management States of Link and Components §

| Downstream Component D-State | Permissible Upstream Component D-State | Permissible Interconnect State |
|------------------------------|--|--|
| <u>D0</u> | <u>D0</u> | <u>L0</u> , <u>L0s</u> , <u>L1</u> ⁽¹⁾ , <u>L2/L3 Ready</u> |
| <u>D1</u> | <u>D0 - D1</u> | <u>L1</u> , <u>L2/L3 Ready</u> |
| <u>D2</u> | <u>D0 - D2</u> | <u>L1</u> , <u>L2/L3 Ready</u> |
| <u>D3Hot</u> | <u>D0 - D3Hot</u> | <u>L1</u> , <u>L2/L3 Ready</u> |
| <u>D3Cold</u> | <u>D0 - D3Cold</u> | <u>L2</u> ⁽²⁾ , <u>L3</u> |

Notes:

1. Requirements for ASPM L0s and ASPM L1 support are form factor specific.
2. If Vaux is provided by the platform, the Link sleeps in L2. In the absence of Vaux, the L-state is L3.

The following rules relate to PCI-PM compatible power management:

- Devices in D0 , D1 , D2 , and D3Hot must respond to the receipt of a PME_Turn_Off Message by the transmission of a PME_TO_Ack Message.
- In any device D state, following the execution of a PME_Turn_Off / PME_TO_Ack handshake sequence, a Downstream component must request a Link transition to L2/L3 Ready using the PM_Enter_L23 DLLP. Following the L2/L3 Ready entry transition protocol the Downstream component must be ready for loss of main power and reference clock.
- The Upstream Port of a Single-Function Device must initiate a Link state transition to L1 based solely upon its Function being programmed to D1 , D2 , or D3Hot . In the case of the Switch, system software bears the responsibility of ensuring that any D-state programming of a Switch's Upstream Port is done in a compliant manner with respect to hierarchy-wide PM policies (i.e., the Upstream Port cannot be programmed to a D-state that is any less active than the most active Downstream Port and Downstream connected component/ Function(s)).
- The Upstream Port of a non-ARI Multi-Function Device must not initiate a Link state transition to L1 (on behalf of PCI-PM) until all of its Functions have been programmed to a non-D0 D-state.

- The Upstream Port of an ARI Device must not initiate a Link state transition to L1 (on behalf of PCI-PM) until at least one of its Functions has been programmed to a non-D0 state, and all of its Functions are either in a non-D0 state or the D0_{uninitialized} State.
- With ~~↓↓SR-IOV devices,↑ ↑↑SR-IOV Devices,↑~~ the Link Power State is controlled solely by the setting in the PFs, regardless of the VFs' D-states. VF Power States do not affect the Link Power State.
- ~~↑↑With SIOV devices , the Link Power State is controlled solely by the setting in the PFs. SDIs do not have D-states.↑~~

ECN: Base 6.3
SIOV△↔

5.3.2.1 Entry into the L1 State §

§ Figure 5-2 depicts the process by which a Link transitions into the L1 state as a direct result of power management software programming the Downstream connected component into a lower power state, (either D1, D2, or D3_{Hot} state). This figure and the subsequent description outline the transition process for a single -Function Downstream component that is being programmed to a non-D0 state.

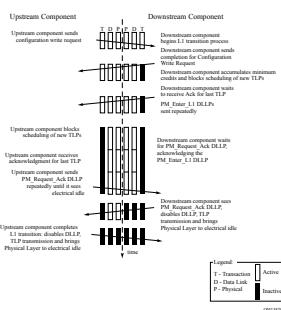


Figure 5-2 Entry into the L1 Link State §

The following text provides additional detail for the Link state transition process shown in § Figure 5-2 .

PM Software Request:

- PM software sends a Configuration Write Request TLP to the Downstream Function's PMCSR to change the Downstream Function's D-state (from D0 to D1 for example).

Downstream Component Link State Transition Initiation Process:

- The Downstream component schedules the Completion corresponding to the Configuration Write Request to its PMCSR PowerState field and accounts for the completion credits required.
- The Downstream component must then wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type for all enabled VCs (if it does not already have such credits). All Transaction Layer TLP scheduling is then suspended.
- The Downstream component then waits until it receives an acknowledgement for the PMCSR Write Completion, and any other TLPs it had previously sent. The component must retransmit a TLP out of its appropriate Retry Buffer if required to do so by the Data Link Layer rules (when operating in Non Flit Mode) or the Flit Ack/Nak rules (when operating in Flit Mode).
- Once all of the Downstream components' TLPs have been acknowledged, the Downstream component starts to transmit PM_Enter_L1 DLLPs. The Downstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Enter_L1 DLLP, in Non-Flit Mode. The transmission of other DLLPs and

SKP Ordered Sets is permitted at any time between PM_Enter_L1 transmissions, and do not contribute to this idle time limit.

The Downstream component continues to transmit the PM_Enter_L1 DLLP as described above until it receives a response from the Upstream component¹⁰⁶ (PM_Request_Ack).

The Downstream component must continue to accept TLPs and DLLPs from the Upstream component, and continue to respond with DLLPs, including FC update DLLPs and Ack/Nak DLLPs, as required. Any TLPs that are blocked from transmission (including responses to TLP(s) received) must be stored for later transmission, and must cause the Downstream component to initiate L1 exit as soon as possible following L1 entry.

Upstream Component Link State Transition Process:

6. Upon receiving the PM_Enter_L1 DLLP, the Upstream component blocks the scheduling of all TLP transmissions.
7. The Upstream component then must wait until it receives an acknowledgement for the last TLP it had previously sent. The Upstream component must retransmit a TLP from its appropriate Retry Buffer if required to do so by the Data Link Layer rules (when operating in Non Flit Mode) or the Flit Ack/Nak rules (when operating in Flit Mode).
8. Once all of the Upstream component's TLPs have been acknowledged, the Upstream component must send PM_Request_Ack DLLPs Downstream, regardless of any outstanding Requests. The Upstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Request_Ack DLLP, in Non-Flit Mode. The transmission of SKP Ordered Sets is permitted at any time between PM_Request_Ack transmissions, and does not contribute to this idle time limit.

The Upstream component continues to transmit the PM_Request_Ack DLLP as described above until it observes its receive Lanes enter into the Electrical Idle state. Refer to § Chapter 4. for more details on the Physical Layer behavior.

Completing the L1 Link State Transition:

9. Once the Downstream component has captured the PM_Request_Ack DLLP on its Receive Lanes (signaling that the Upstream component acknowledged the transition to L1 request), it then disables DLLP transmission and brings the Upstream directed physical Link into the Electrical Idle state.
10. When the Receive Lanes on the Upstream component enter the Electrical Idle state, the Upstream component stops sending PM_Request_Ack DLLPs, disables DLLP transmission, and brings its Transmit Lanes to Electrical Idle completing the transition of the Link to L1.

When two components' interconnecting Link is in L1 as a result of the Downstream component being programmed to a non-D0 state, both components suspend the operation of their Flow Control Update and, if implemented, UpdateFC FCP Timer (see § Section 2.6.1.2) counter mechanisms. Refer to § Chapter 4. for more detail on the Physical Layer behavior.

Refer to § Section 5.2 if the negotiation to L1 is interrupted.

Components on either end of a Link in L1 may optionally disable their internal PLLs in order to conserve more energy. Note, however, that platform supplied main power and reference clocks must continue to be supplied to components on both ends of an L1 Link in the L1.0 substate of L1 .

Refer to § Section 5.5 for entry into the L1 PM Substates .

¹⁰⁶. If at this point the Downstream component needs to initiate a transfer on the Link, it must first complete the transition to L1 . Once in L1 it is then permitted to initiate an exit L1 to handle the transfer.

Base 6.4 vs Base 6.3

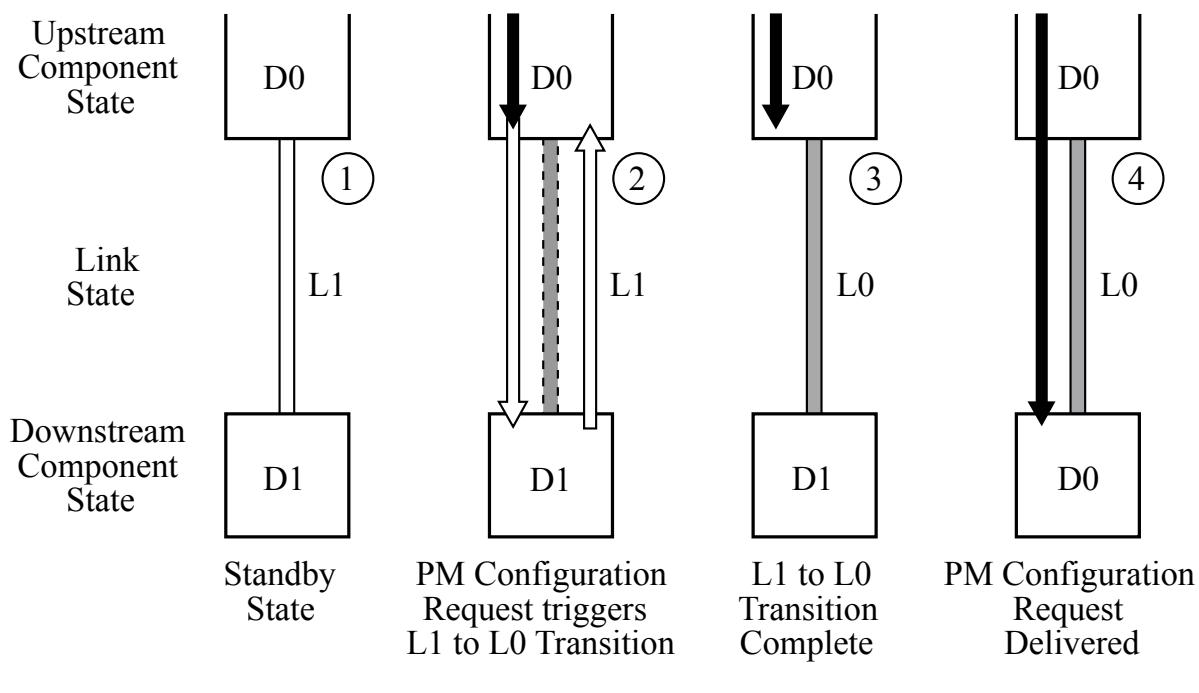
5.3.2.2 Exit from L1 State §

L1 exit can be initiated by the component on either end of a Link.

Upon exit from L1, it is recommended that the Downstream component send flow control update DLLPs for all enabled VCs and FC types starting within 1 µs of L1 exit.

The physical mechanism for transitioning a Link from L1 to L0 is described in detail in § Chapter 4..

L1 exit must be initiated by a component if that component needs to transmit a TLP on the Link. An Upstream component must initiate L1 exit on a Downstream Port even if it does not have the flow control credits needed to transmit the TLP that it needs to transmit. Following L1 exit, the Upstream component must wait to receive the needed credit from the Downstream component. § Figure 5-3 outlines an example sequence that would trigger an Upstream component to initiate transition of the Link to the L0 state.



OM13821

Figure 5-3 Exit from L1 Link State Initiated by Upstream Component §

Sequence of events:

1. Power management software initiates a configuration cycle targeting a PM configuration register (the PowerState field of the PMCSR in this example) within a Function that resides in the Downstream component (e.g., to bring the Function back to the D0 state).
2. The Upstream component detects that a configuration cycle is intended for a Link that is currently in a low power state, and as a result, initiates a transition of that Link into the L0 state.
3. If the Link is in either L1.1 or L1.2 substates of L1, then the Upstream component initiates a transition of the Link into the L1.0 substate of L1.
4. In accordance with the § Chapter 4. definition, both directions of the Link enter into Link training, resulting in the transition of the Link to the L0 state. The L1 → L0 transition is discussed in detail in § Chapter 4..

5. Once both directions of the Link are back to the active L0 state, the Upstream Port sends the configuration Packet Downstream.

5.3.2.3 Entry into the L2/L3 Ready State §

Transition to the L2/L3 Ready state follows a process that is similar to the L1 entry process. There are some minor differences between the two that are spelled out below.

- L2/L3 Ready entry transition protocol does not immediately result in an L2 or L3 Link state. The transition to L2/L3 Ready is effectively a handshake to establish the Downstream component's readiness for power removal. L2 or L3 is ultimately achieved when the platform removes the components' power and reference clock.
- The time for L2/L3 Ready entry transition is indicated by the completion of the PME_Turn_Off / PME_TO_Ack handshake sequence. Any actions on the part of the Downstream component necessary to ready itself for loss of power must be completed prior to initiating the transition to L2/L3 Ready. Once all preparations for loss of power and clock are completed, L2/L3 Ready entry is initiated by the Downstream component by sending the PM_Enter_L23 DLLP Upstream.
- L2/L3 Ready entry transition protocol uses the PM_Enter_L23 DLLP.

Note that the PM_Enter_L23 DLLPs are sent continuously until an acknowledgement is received or power is removed.

- Refer to § Section 5.2 if the negotiation to L2/L3 Ready is interrupted.

Base 6.4 vs Base 6.3

5.3.3 Power Management Event Mechanisms §

5.3.3.1 Motivation §

The PCI Express PME mechanism is software compatible with the [PCI] PME mechanism. Power Management Events are generated by Functions as a means of requesting a PM state change. Power Management Events are typically utilized to revive the system or an individual Function from a low power state.

Power management software may transition a Hierarchy into a low power state, and transition the Upstream Links of these devices into the non-communicating L2 state.¹⁰⁷ The PCI Express PME generation mechanism is, therefore, broken into two components:

- Waking a non-communicating Hierarchy (wakeup). This step is required only if the Upstream Link of the device originating the PME is in the non-communicating L2 state, since in that state the device cannot send a PM_PME Message Upstream.
- Sending a PM_PME Message to the root of the Hierarchy

PME indications that originate from PCI Express Endpoints or PCI Express Legacy Endpoints are propagated to the Root Complex in the form of TLP messages. PM_PME Messages identify the requesting agent within the Hierarchy (via the Requester ID of the PM_PME Message header). Explicit identification within the PM_PME Message is intended to facilitate quicker PME service routine response, and hence shorter resume time.

If an RCIEP is associated with a Root Complex Event Collector, any PME indications that originate from that RCIEP must be reported by that Root Complex Event Collector.

PME indications that originate from a Root Port itself are reported through the same Root Port.

¹⁰⁷. The L2 state is defined as “non-communicating” since component reference clock and main power supply are removed in that state.

5.3.3.2 Link Wakeup §

The Link wakeup mechanisms provide a means of signaling the platform to re-establish power and reference clocks to the components within its domain. There are two defined wakeup mechanisms: Beacon and WAKE#. The Beacon mechanism uses in-band signaling to implement wakeup functionality. For components that support wakeup functionality, the form factor specification(s) targeted by the implementation determine the support requirements for the wakeup mechanism. Switch components targeting applications where Beacon is used on some Ports of the Switch and WAKE# is used for other Ports must translate the wakeup mechanism appropriately (see the implementation note [Example of WAKE# to Beacon Translation in § Section 5.3.3.2](#)). In applications where WAKE# is the only wakeup mechanism used, the Root Complex is not required to support the receipt of Beacon.

The WAKE# mechanism uses sideband signaling to implement wakeup functionality. WAKE# is an “open drain” signal asserted by components requesting wakeup and observed by the associated power controller. WAKE# is only defined for certain form factors, and the detailed specifications for WAKE# are included in the relevant form factor specifications. Specific form factor specifications may require the use of either Beacon or WAKE# as the wakeup mechanism.

When WAKE# is used as a wakeup mechanism, once WAKE# has been asserted, the asserting Function must continue to drive the signal low until main power has been restored to the component as indicated by Fundamental Reset going inactive.

The system is not required to route or buffer WAKE# in such a way that an Endpoint is guaranteed to be able to detect that the signal has been asserted by another Function.

Before using any wakeup mechanism, a Function must be enabled by software to do so by setting the Function's PME_En bit in the PMCSR. The PME_Status bit is sticky, and Functions must maintain the value of the PME_Status bit through reset if auxiliary power is available and they are enabled for wakeup events (this requirement also applies to the PME_En bit in the PMCSR and the Aux Power PM Enable bit in the Device Control Register).

Systems that allow PME generation from D3Cold state must provide auxiliary power to support Link wakeup when the main system power rails are off. A component may only consume auxiliary power if software has enabled it to do so as described in § Section 5.6. Software is required to enable auxiliary power consumption in all components that participate in Link wakeup, including all components that must propagate the Beacon signal. In the presence of legacy system software, this is the responsibility of system firmware.

Regardless of the wakeup mechanism used, once the Link has been re-activated and trained, the requesting agent then propagates a PM_PME Message Upstream to the Root Complex. From a power management point of view, the two wakeup mechanisms provide the same functionality, and are not distinguished elsewhere in this chapter.

IMPLEMENTATION NOTE: EXAMPLE OF WAKE# TO BEACON TRANSLATION §

Switch components targeting applications that connect “Beacon domains” and “WAKE# domains” must translate the wakeup mechanism appropriately. § Figure 5-4 shows two example systems, each including slots that use the WAKE# wakeup mechanism. In Case 1, WAKE# is input directly to the Power Management Controller, and no translation is required. In Case 2, WAKE# is an input to the Switch, and in response to WAKE# being asserted the Switch must generate a Beacon that is propagated to the Root Complex/Power Management Controller.

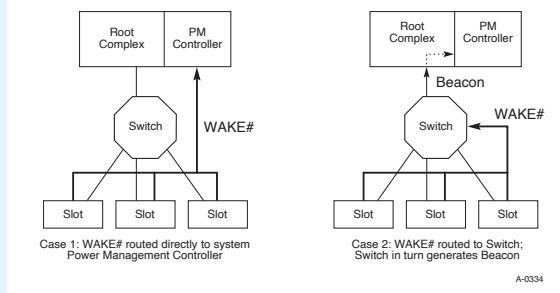


Figure 5-4 Conceptual Diagrams Showing Two Example Cases of WAKE# Routing §

5.3.3.2.1 PME Synchronization §

PCI Express-PM introduces a fence mechanism that serves to initiate the power removal sequence while also coordinating the behavior of the platform's power management controller and PME handling by PCI Express agents.

PME_Turn_Off Broadcast Message

Before main component power and reference clocks are turned off, the Root Complex or Switch Downstream Port must issue a broadcast Message that instructs all agents Downstream of that point within the hierarchy to cease initiation of any subsequent PM_PME Messages, effective immediately upon receipt of the PME_Turn_Off Message.

Each PCI Express agent is required to respond with a TLP “acknowledgement” Message, PME_TO_Ack that is always routed Upstream. In all cases, the PME_TO_Ack Message must terminate at the PME_Turn_Off Message's point of origin.¹⁰⁸

A Switch must report an “aggregate” acknowledgement only after having received PME_TO_Ack Messages from each of its Downstream Ports. Once a PME_TO_Ack Message has arrived on each Downstream Port, the Switch must then send a PME_TO_Ack packet on its Upstream Port. The occurrence of any one of the following must reset the aggregation mechanism: the transmission of the PME_TO_Ack Message from the Upstream Port, the receipt of any TLP at the Upstream Port, the removal of main power to the Switch, or Fundamental Reset.

All components with an Upstream Port must accept and acknowledge the PME_Turn_Off Message regardless of the D state of the associated device or any of its Functions for a Multi-Function Device. Once a component has sent a PME_TO_Ack Message, it must then prepare for removal of its power and reference clocks by initiating a transition to the L2/L3 Ready state.

^{108.} Point of origin for the PME_Turn_Off Message could be all of the Root Ports for a given Root Complex (full platform sleep state transition), an individual Root Port, or a Switch Downstream Port.

A Switch must transition its Upstream Link to the L2/L3 Ready state after all of its Downstream Ports have entered the L2/L3 Ready state.

The Links attached to the originator of the PME_Turn_Off Message are the last to assume the L2/L3 Ready state. This state transition serves as an indication to the power delivery manager¹⁰⁹ that all Links within that portion of the Hierarchy have successfully retired all in flight PME Messages to the point of PME_Turn_Off Message origin and have performed any necessary local conditioning in preparation for power removal.

In order to avoid deadlock in the case where one or more devices do not respond with a PME_TO_Ack Message and then put their Links into the L2/L3 Ready state, the power manager must implement a timeout after waiting for a certain amount of time, after which it proceeds as if the Message had been received and all Links put into the L2/L3 Ready state. The recommended limit for this timer is in the range of 1 ms to 10 ms.

The power delivery manager must wait a minimum of 100 ns after observing all Links corresponding to the point of origin of the PME_Turn_Off Message enter L2/L3 Ready before removing the components' reference clock and main power. This requirement does not apply in the case where the above mentioned timer triggers.

IMPLEMENTATION NOTE: PME_TO_ACK MESSAGE PROXY BY SWITCHES §

One of the PME_Turn_Off / PME_TO_Ack handshake's key roles is to ensure that all in flight PME Messages are flushed from the PCI Express fabric prior to sleep state power removal. This is guaranteed to occur because PME Messages and the PME_TO_Ack Messages both use the posted request queue within VC0 and so all previously injected PME Messages will be made visible to the system before the PME_TO_Ack is received at the Root Complex. Once all Downstream Ports of the Root Complex receive a PME_TO_Ack Message the Root Complex can then signal the power manager that it is safe to remove power without loss of any PME Messages.

Switches create points of hierarchical expansion and, therefore, must wait for all of their connected Downstream Ports to receive a PME_TO_Ack Message before they can send a PME_TO_Ack Message Upstream on behalf of the sub-hierarchy that it has created Downstream. This can be accomplished very simply using common score boarding techniques. For example, once a PME_Turn_Off broadcast Message has been broadcast Downstream of the Switch, the Switch simply checks off each Downstream Port having received a PME_TO_Ack. Once the last of its active Downstream Ports receives a PME_TO_Ack, the Switch will then send a single PME_TO_Ack Message Upstream as a proxy on behalf of the entire sub-hierarchy Downstream of it. Note that once a Downstream Port receives a PME_TO_Ack Message and the Switch has scored its arrival, the Port is then free to drop the packet from its internal queues and free up the corresponding posted request queue FC credits.

5.3.3.3 PM_PME Messages §

PM_PME Messages are posted Transaction Layer Packets (TLPs) that inform the power management software which agent within the Hierarchy requests a PM state change. PM_PME Messages, like all other Power Management system Messages, must use the general purpose Traffic Class, TC0.

PM_PME Messages are always routed in the direction of the Root Complex. To send a PM_PME Message on its Upstream Link, a device must transition the Link to the L0 state (if the Link was not in that state already). Unless otherwise noted, the device will keep the Link in the L0 state following the transmission of a PM_PME Message .

¹⁰⁹. Power delivery control within this context relates to control over the entire Link hierarchy, or over a subset of Links ranging down to a single Link and associated Endpoint for sub hierarchies supporting independently managed power and clock distribution.

5.3.3.3.1 PM_PME “Backpressure” Deadlock Avoidance §

A Root Complex is typically implemented with local buffering to store temporarily a finite number of PM_PME Messages that could potentially be simultaneously propagating through the Hierarchy. Given a limited number of PM_PME Messages that can be stored within the Root Complex, there can be backpressure applied to the Upstream directed posted queue in the event that the capacity of this temporary PM_PME Message buffer is exceeded.

Deadlock can occur according to the following example scenario:

1. Incoming PM_PME Messages fill the Root Complex's temporary storage to its capacity while there are additional PM_PME Messages still in the Hierarchy making their way Upstream.
2. The Root Complex, on behalf of system software, issues a Configuration Read Request targeting one of the PME requester's PMCSR (e.g., reading its PME_Status bit).
3. The corresponding split completion Packet is required, as per producer/consumer ordering rules, to push all previously posted PM_PME Messages ahead of it, which in this case are PM_PME Messages that have no place to go.
4. The PME service routine cannot make progress; the PM_PME Message storage situation does not improve.
5. Deadlock occurs.

Precluding potential deadlocks requires the Root Complex to always enable forward progress under these circumstances. This must be done by accepting any PM_PME Messages that posted queue flow control credits allow for, and discarding any PM_PME Messages that create an overflow condition. This required behavior ensures that no deadlock will occur in these cases; however, PM_PME Messages will be discarded and hence lost in the process.

To ensure that no PM_PME Messages are lost permanently, all agents that are capable of generating PM_PME must implement a PME Service Timeout mechanism to ensure that their PME requests are serviced in a reasonable amount of time.

If after 100 ms (+50%/-5%), the PME_Status bit of a requesting agent has not yet been cleared, the PME Service Timeout mechanism expires triggering the PME requesting agent to re-send the temporarily lost PM_PME Message . If at this time the Link is in a non-communicating state, then, prior to re-sending the PM_PME Message , the agent must reactivate the Link as defined in § Section 5.3.3.2 .

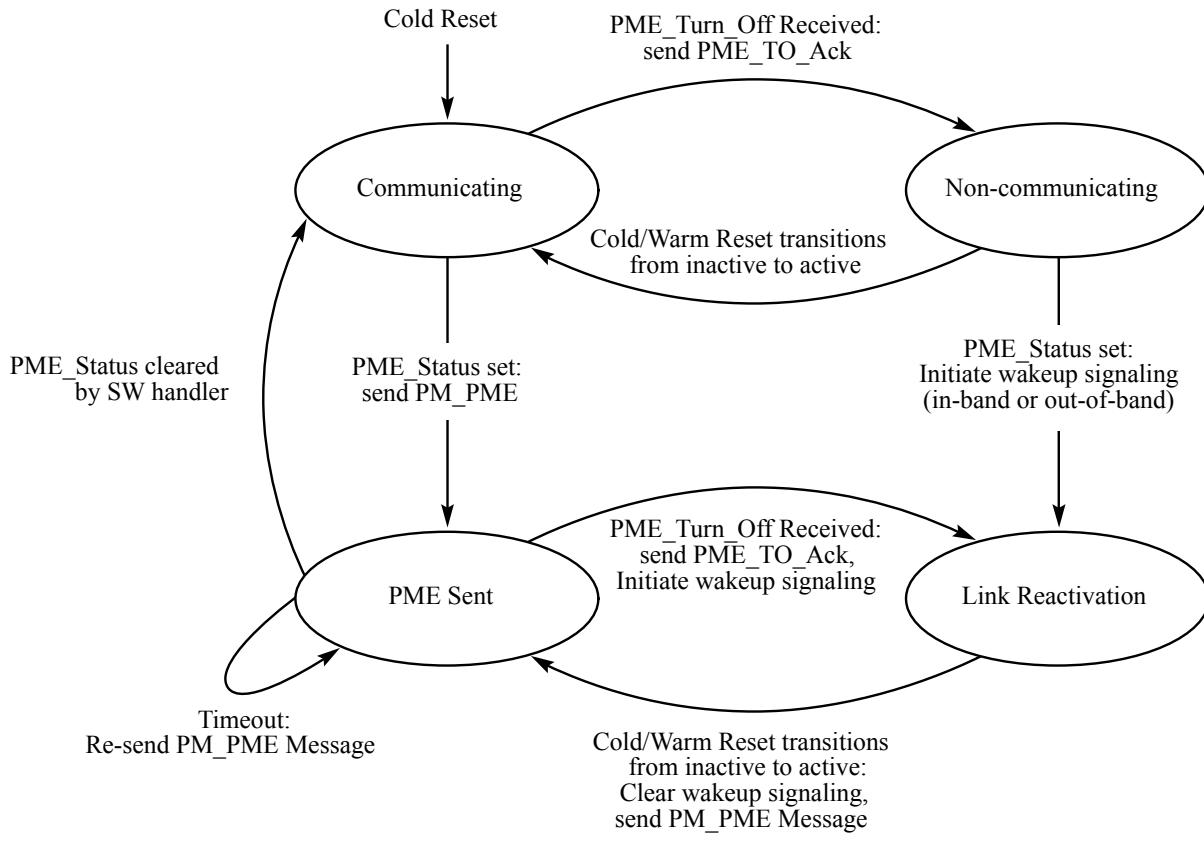
5.3.3.4 PME Rules §

- All device Functions must implement the PCI-PM Power Management Capabilities (PMC) register and the PMCSR in accordance with the PCI-PM specification. These registers reside in the PCI-PM compliant PCI Capability List format.
 - PME capable Functions must implement the PME_Status bit, and underlying functional behavior, in their PMCSR.
 - When a Function initiates Link wakeup, or issues a PM_PME Message , it must set its PME_Status bit.
- Switches must route a PM_PME received on any Downstream Port to their Upstream Port
- On receiving a PME_Turn_Off Message, the device must block the transmission of PM_PME Messages and transmit a PME_TO_Ack Message Upstream. The component is permitted to send a PM_PME Message after the Link is returned to an L0 state through LDn .
- Before a Link or a portion of a Hierarchy is transferred into a non-communicating state (i.e., a state from which it cannot issue a PM_PME Message), a PME_Turn_Off Message must be broadcast Downstream.

Base 6.4 vs Base 6.3

5.3.3.5 PM_PME Delivery State Machine §

The following diagram conceptually outlines the PM_PME delivery control state machine. This state machine determines the ability of a Link to service PME events by issuing PM_PME immediately vs. requiring Link wakeup.



OM13822A

Figure 5-5 A Conceptual PME Control State Machine §

Communicating State:

At initial power-up and associated reset, the Upstream Link enters the Communicating state

- If PME_Status is asserted (assuming PME delivery is enabled), a PM_PME Message will be issued Upstream, terminating at the root of the Hierarchy. The next state is the PME Sent state
- If a PME_Turn_Off Message is received, the Link enters the Non-communicating state following its acknowledgment of the Message and subsequent entry into the L2/L3 Ready state.

Non-communicating State:

- Following the restoration of power and clock, and the associated reset, the next state is the Communicating state.
- If PME_Status is asserted, the Link will transition to the Link Reactivation state, and activate the wakeup mechanism.

PME Sent State

- If PME_Status is cleared, the Function becomes PME Capable again. Next state is the Communicating state.
- If the PME_Status bit is not Clear by the time the PME service timeout expires, a PM_PME Message is re-sent Upstream. Refer to § Section 5.3.3.3.1 for an explanation of the timeout mechanism.
- If a PME Message has been issued but the PME_Status has not been cleared by software when the Link is about to be transitioned into a messaging incapable state (a PME_Turn_Off Message is received), the Link transitions into Link Reactivation state after sending a PME_TO_Ack Message. The device also activates the wakeup mechanism.

Link Reactivation State

- Following the restoration of power and clock, and the associated reset, the Link resumes a transaction-capable state. The device clears the wakeup signaling, if necessary, and issues a PM_PME Upstream and transitions into the PME Sent state.

5.4 Native PCI Express Power Management Mechanisms §

The following sections define power management features that require new software. While the presence of these features in new PCI Express designs will not break legacy software compatibility, taking the full advantage of them requires new code to manage them.

These features are enumerated and configured using PCI Express native configuration mechanisms as described in § Chapter 7. of this specification. Refer to § Chapter 7. for specific register locations, bit assignments, and access mechanisms associated with these PCI Express-PM features.

5.4.1 Active State Power Management (ASPM) §

All Ports not associated with an Internal Root Complex Link or system Egress Port are required to support the minimum requirements defined herein for Active State Link PM. This feature must be treated as being orthogonal to the PCI-PM software compatible features from a minimum requirements perspective. For example, the Root Complex is exempt from the PCI-PM software compatible features requirements; however, it must implement the minimum requirements of ASPM.

Components in the D0 state (i.e., fully active state) normally keep their Upstream Link in the active L0 state, as defined in § Section 5.3.2. ASPM defines a protocol for components in the D0 state to reduce Link power by placing their Links into a low power state and instructing the other end of the Link to do likewise. This capability allows hardware-autonomous, dynamic Link power reduction beyond what is achievable by software-only controlled (i.e., PCI-PM software driven) power management.

In Non-Flit Mode there are two low power "standby" Link states defined for ASPM, L0s and L1. In Flit Mode L0p effectively replaces L0s, and L1 remains as a "standby" Link state for ASPM.

The L0s low power Link state is optimized for short entry and exit latencies, while providing substantial power savings. If the L0s state is enabled in a device, it is recommended that the device bring its Transmit Link into the L0s state whenever that Link is not in use (refer to § Section 5.4.1.1 for details relating to the L0s invocation policy). Component support of the L0s Link state from within the D0 device state is optional unless the applicable form factor specification for the Link explicitly requires it.

The L0p low power Link state is optimized for short entry and longer exit latencies, while providing substantial power savings and supporting Link operation while a Link width change is in progress.

The L1 Link state is optimized for maximum power savings at a cost of longer entry and exit latencies. L1 reduces Link power beyond the L0s state for cases where very low power is required and longer transition times are acceptable. ASPM support for the L1 Link state is optional unless specifically required by a particular form factor.

Optional L1 PM Substates L1.1 and L1.2 are defined. These substates can further reduce Link power for cases where very low idle power is required, and longer transition times are acceptable.

Each component must report its level of support for ASPM in the ASPM Support field. As applicable, each component shall also report its L0s and L1 exit latency (the time that it requires to transition from the L0s or L1 state to the L0 state). Endpoint Functions must also report the worst-case latency that they can withstand before risking, for example, internal FIFO overruns due to the transition latency from L0s or L1 to the L0 state. Power management software can use the provided information to then enable the appropriate level of ASPM.

The L1 exit latency also applies to L0p, but when used for L0p, indicates the time required to widen the Link. The Link remains operational during this time period, but at lower bandwidth.

NOTE: L0p and ASPM §

A future draft of this specification may define a mechanism to report the worst case latency an Endpoint can withstand at L0p reduced bandwidth. This may involve multiple latency requirement values depending on the beginning and ending Link widths. Power management software could use this information to enable appropriate L0p Link widths for ASPM.

The L0s exit latency may differ significantly if the reference clock for opposing sides of a given Link is provided from the same source, or delivered to each component from a different source. PCI Express-PM software informs each device of its clock configuration via the Common Clock Configuration bit in its Capability structure's Link Control register. This bit serves as the determining factor in the L0s exit latency value reported by the device. ASPM may be enabled or disabled by default depending on implementation specific criteria and/or the requirements of the associated form factor specification(s). Software can enable or disable ASPM using a process described in § Section 5.4.1.4.1.

Power management software enables or disables ASPM in each Port of a component by programming the ASPM Control field. Note that new BIOS code can effectively enable or disable ASPM functionality when running with a legacy operating system, but a PCI Express-aware operating system might choose to override ASPM settings configured by the BIOS.

IMPLEMENTATION NOTE: ISOCHRONOUS TRAFFIC AND ASPM §

Isochronous traffic requires bounded service latency. ASPM may add latency to isochronous transactions beyond expected limits. A possible solution would be to disable ASPM for devices that are configured with an Isochronous Virtual Channel.

For ARI Devices, ASPM Control is determined solely by the setting in Function 0, regardless of Function 0's D-state. The ASPM Control settings in other Functions are ignored by the component.

An Upstream Port of a non-ARI Multi-Function Device may be programmed with different values in their respective ASPM Control fields of each Function. The policy for such a component will be dictated by the most active common denominator among all D0 Functions according to the following rules:

- Functions in a non-D0 state (D1 and deeper) are ignored in determining the ASPM policy

- If any of the Functions in the D0 state has its ASPM disabled (ASPM Control field = 00b) or if at least one of the Functions in the D0 state is enabled for L0s only (ASPM Control field = 01b) and at least one other Function in the D0 state is enabled for L1 only (ASPM Control field = 10b), then ASPM is disabled for the entire component
- Else, if at least one of the Functions in the D0 state is enabled for L0s only (ASPM Control field = 01b), then ASPM is enabled for L0s only
- Else, if at least one of the Functions in the D0 state is enabled for L1 only (ASPM Control field = 10b), then ASPM is enabled for L1 only
- Else, ASPM is enabled for both L0s and L1 states

Note that the components must be capable of changing their behavior during runtime as device Functions enter and exit low power device states. For example, if one Function within a Multi-Function Device is programmed to disable ASPM, then ASPM must be disabled for that device while that Function is in the D0 state. Once the Function transitions to a non-D0 state, ASPM can be enabled if all other Functions are enabled for ASPM.

5.4.1.1 L0s ASPM State §

IMPLEMENTATION NOTE: L0S ONLY WORKS IN NON-FLIT MODE WITH NO RETIMERS §

Flit Mode does not support L0s .

Retimers do not support L0s .

Device support of the L0s low power Link state is optional unless the applicable form factor specification for the Link explicitly requires it.

IMPLEMENTATION NOTE: POTENTIAL ISSUES WITH LEGACY SOFTWARE WHEN L0S IS NOT SUPPORTED §

In earlier versions of this specification, device support of L0s was mandatory, and software could legitimately assume that all devices support L0s. Newer hardware components that do not support L0s may encounter issues with such “legacy software”. Such software might not even check the ASPM Support field in the Link Capabilities register, might not recognize the subsequently defined values (00b and 10b) for the ASPM Support field, or might not follow the policy of enabling L0s only if components on both sides of the Link each support L0s.

Legacy software (either operating system or firmware) that encounters the previously reserved value 00b (No ASPM Support), will most likely refrain from enabling L1, which is intended behavior. Legacy software will also most likely refrain from enabling L0s for that component's Transmitter (also intended behavior), but it is unclear if such software will also refrain from enabling L0s for the component on the other side of the Link. If software enables L0s on one side when the component on the other side does not indicate that it supports L0s, the result is undefined. Situations where the resulting behavior is unacceptable may need to be handled by updating the legacy software, establishing a list of configurations for which the legacy software is directed not to enable L0s, or simply not supporting the problematic system configurations.

On some platforms, firmware controls ASPM, and the operating system may either preserve or override the ASPM settings established by firmware. This will be influenced by whether the operating system supports controlling ASPM, and in some cases by whether the firmware permits the operating system to take control of ASPM. Also, ASPM control with hot-plug operations may be influenced by whether native PCI Express hot-plug versus ACPI hot-plug is used. Addressing any legacy software issues with L0s may require updating the firmware, the operating system, or both.

When a component does not advertise that it supports L0s, as indicated by its ASPM Support field value being 00b or 10b, it is recommended that the component's L0s Exit Latency field return a value of 111b, indicating the maximum latency range. Advertising this maximum latency range may help discourage legacy software from enabling L0s if it otherwise would do so, and thus may help avoid problems caused by legacy software mistakenly enabling L0s on this component or the component on the other side of the Link.

Transaction Layer and Link Layer timers are not affected by a transition to the L0s state (i.e., they must follow the rules as defined in their respective chapters).

IMPLEMENTATION NOTE: MINIMIZING L0S EXIT LATENCY §

L0s exit latency depends mainly on the ability of the Receiver to quickly acquire bit and Symbol synchronization. Different approaches exist for high-frequency clocking solutions which may differ significantly in their L0s exit latency, and therefore in the efficiency of ASPM. To achieve maximum power savings efficiency with ASPM, L0s exit latency should be kept low by proper selection of the clocking solution.

5.4.1.1.1 Entry into the L0s State §

Entry into the L0s state is managed separately for each direction of the Link. It is the responsibility of each device at either end of the Link to initiate an entry into the L0s state on its transmitting Lanes. Software must not enable L0s in either direction on a given Link unless components on both sides of the Link each support L0s; otherwise, the result is undefined.

A Port that is disabled for the L0s state must not transition its transmitting Lanes to the L0s state. However, if the Port advertises that it supports L0s, Port must be able to tolerate having its Receiver Port Lanes enter L0s, (as a result of the device at the other end bringing its transmitting Lanes into L0s state), and then later returning to the L0 state.

L0s Invocation Policy

Ports that are enabled for L0s entry generally should transition their Transmit Lanes to the L0s state if the defined idle conditions (below) are met for a period of time, recommended not to exceed 7 µs. Within this time period, the policy used by the Port to determine when to enter L0s is implementation specific. It is never mandatory for a Transmitter to enter L0s.

Definition of Idle

The definition of an “idle” Upstream Port varies with device Function category. An Upstream Port of a Multi-Function Device is considered idle only when all of its Functions are idle.

A non-Switch Port is determined to be idle if the following conditions are met:

- No TLP is pending to transmit over the Link, or no FC credits are available to transmit any TLPs
- No DLLPs are pending for transmission

A Switch Upstream Port Function is determined to be idle if the following conditions are met:

- None of the Switch's Downstream Port Receive Lanes are in the L0, Recovery, or Configuration state
- No pending TLPs to transmit, or no FC credits are available to transmit anything
- No DLLPs are pending for transmission

A Switch's Downstream Port is determined to be idle if the following conditions are met:

- The Switch's Upstream Port's Receive Lanes are not in the L0, Recovery, or Configuration state
- No pending TLPs to transmit on this Link, or no FC credits are available
- No DLLPs are pending for transmission

Refer to § Section 4.2 for details on L0s entry by the Physical Layer.

5.4.1.1.2 Exit from the L0s State §

A component with its Transmitter in L0s must initiate L0s exit when it has a TLP or DLLP to transmit across the Link. Note that a transition from the L0s Link state does not depend on the status (or availability) of FC credits. The Link must be able to reach the L0 state, and to exchange FC credits across the Link. For example, if all credits of some type were consumed when the Link entered L0s, then any component on either side of the Link must still be able to transition the Link to the L0 state when new credits need to be sent across the Link. Note that it may be appropriate for a component to anticipate the end of the idle condition and initiate L0s transmit exit; for example, when an NP request is received.

Downstream Initiated Exit

The Upstream Port of a component is permitted to initiate an exit from the L0s low-power state on its Transmit Link, (Upstream Port Transmit Lanes in the case of a Downstream Switch), if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the Upstream direction as described in § [Section 4.2](#).

If the Upstream component is a Switch (i.e., it is not the Root Complex), then it must initiate a transition on its Upstream Port Transmit Lanes (if the Upstream Port's Transmit Lanes are in a low-power state) as soon as it detects an exit from L0s on any of its Downstream Ports.

Upstream Initiated Exit

A Downstream Port is permitted to initiate an exit from L0s low power state on any of its Transmit Links if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the Downstream direction as described in § [Chapter 4](#).

If the Downstream component contains a Switch, it must initiate a transition on all of its Downstream Port Transmit Lanes that are in L0s at that time as soon as it detects an exit from L0s on its Upstream Port. Links that are already in the L0 state are not affected by this transition. Links whose Downstream component is in a low-power state (i.e., D1 - D3_{Hot} states) are also not affected by the exit transitions.

For example, consider a Switch with an Upstream Port in L0s and a Downstream device in a D1 state. A configuration request packet travels Downstream to the Switch, intending ultimately to reprogram the Downstream device from D1 to D0. The Switch's Upstream Port Link must transition to the L0 state to allow the packet to reach the Switch. The Downstream Link connecting to the device in D1 state will not transition to the L0 state yet; it will remain in the L1 state. The captured packet is checked and routed to the Downstream Port that shares a Link with the Downstream device that is in D1. As described in § [Section 4.2](#), the Switch now transitions the Downstream Link to the L0 state. Note that the transition to the L0 state was triggered by the packet being routed to that particular Downstream L1 Link, and not by the transition of the Upstream Port's Link to the L0 state. If the packet's destination was targeting a different Downstream Link, then that particular Downstream Link would have remained in the L1 state.

5.4.1.2 ASPM L0p State §

L0p is a substate of L0 that provides power savings with short entry latency and a longer exit latency. Local L0p exit latency and remote L0p exit latency are visible to software and are reported in the Local L0p Exit Latency and Remote L0p Exit Latency fields of the Data Link Feature Extended Capability.

L0p is supported in Flit Mode only and can be used only when supported by both Link partners. When supported, ASPM L0p is controlled by the Hardware Autonomous Width Disable bit in the Link Control Register and by several Device Control 3 Register fields. See § [Section 4.2.6.7](#) for more detail on L0p.

5.4.1.3 ASPM L1 State §

A component may optionally support the ASPM L1 state; a state that provides greater power savings at the expense of longer exit latency. L1 exit latency is visible to software, and reported via the L1 Exit Latency field.

IMPLEMENTATION NOTE: POTENTIAL ISSUES WITH LEGACY SOFTWARE WHEN ONLY L1 IS SUPPORTED §

In earlier versions of this specification, device support of L0s was mandatory, and there was no architected ASPM Support field value to indicate L1 support without L0s support. Newer hardware components that support only L1 may encounter issues with “legacy software”, i.e., software that does not recognize the subsequently defined value for the ASPM Support field.

Legacy software that encounters the previously reserved value 10b (L1 Support), may refrain from enabling both L0s and L1, which unfortunately avoids using L1 with new components that support only L1. While this may result in additional power being consumed, it should not cause any functional misbehavior. However, the same issues with respect to legacy software enabling L0s exist for this 10b case as are described in the Implementation Note “Potential Issues With Legacy Software When L0s is Not Supported” in § Section 5.4.1.1.

When supported, L1 entry is controlled by the ASPM Control field. Software must enable ASPM L1 on the Downstream component only if it is supported by both components on a Link. Software must sequence the enabling and disabling of ASPM L1 such that the Upstream component is enabled before the Downstream component and disabled after the Downstream component.

5.4.1.3.1 ASPM Entry into the L1 State §

An Upstream Port on a component enabled for L1 ASPM entry may initiate entry into the L1 Link state.

See § Section 5.5.1 for details on transitions into either the L1.1 or L1.2 substates.

IMPLEMENTATION NOTE: INITIATING L1 §

This specification does not dictate when a component with an Upstream Port must initiate a transition to the L1 state. The interoperable mechanisms for transitioning into and out of L1 are defined within this specification; however, the specific ASPM policy governing when to transition into L1 is left to the implementer.

One possible approach would be for the Downstream device to initiate a transition to the L1 state once the device has both its Receiver and Transmitter in the L0s state (RxL0s and TxL0s) for a set amount of time. Another approach would be for the Downstream device to initiate a transition to the L1 state once the Link has been idle in L0 for a set amount of time. This is particularly useful if L0s entry is not enabled. Still another approach would be for the Downstream device to initiate a transition to the L1 state if it has completed its assigned tasks. Note that a component's L1 invocation policy is in no way limited by these few examples.

Three power management Messages provide support for the ASPM L1 state:

- PM_Active_State_Request_L1 (DLLP)
- PM_Request_Ack (DLLP)
- PM_Active_State_Nak (TLP)

Downstream components enabled for ASPM L1 entry negotiate for L1 entry with the Upstream component on the Link.

A Downstream Port must accept a request to enter L1 if all of the following conditions are true:

- The Port supports ASPM L1 entry, and ASPM L1 entry is enabled.¹¹⁰
- No TLP is scheduled for transmission
- No Ack or Nak DLLP is scheduled for transmission (Non-Flit Mode)
- No Flit Ack or Nak is scheduled for transmission (Flit Mode)

A Switch Upstream Port may request L1 entry on its Link provided all of the following conditions are true:

- The Upstream Port supports ASPM L1 entry and it is enabled
- All of the Switch's Downstream Port Links are in the L1 state (or deeper)
- No pending TLPs to transmit
- No pending DLLPs to transmit
- The Upstream Port's Receiver is idle for an implementation specific set amount of time

Note that it is legitimate for a Switch to be enabled for the ASPM L1 Link state on any of its Downstream Ports and to be disabled or not even supportive of ASPM L1 on its Upstream Port. In that case, Downstream Ports may enter the L1 Link state, but the Switch will never initiate an ASPM L1 entry transition on its Upstream Port.

ASPM L1 Negotiation Rules (see § Figure 5-6 and § Figure 5-7):

- In Non-Flit Mode, the Downstream component must not initiate ASPM L1 entry until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type.
- In Flit Mode, for any FC/VC that was initialized with non-zero and non-infinite dedicated credits, the Downstream component must not initiate ASPM L1 entry until it accumulates at least the minimum number of dedicated credits on that VC required to send the largest possible packet for that FC type.
- In Flit Mode, for any FC/VC that was initialized with zero dedicated credits, the Downstream component must not initiate ASPM L1 entry until it accumulates at least the minimum number of shared credits required to send the largest possible packet for that FC type.
- Upon deciding to enter a low-power Link state, the Downstream component must block movement of all TLPs from the Transaction Layer to the Data Link Layer for transmission (including completion packets). If any TLPs become available from the Transaction Layer for transmission during the L1 negotiation process, the transition to L1 must first be completed and then the Downstream component must initiate a return to L0. Refer to § Section 5.2 if the negotiation to L1 is interrupted.
- In Non-Flit Mode, the Downstream component must wait until it receives a Link Layer acknowledgement for the last TLP it had previously sent (i.e., the retry buffer is empty). The component must retransmit a TLP out of its Data Link Layer Retry buffer if required by the Data Link Layer rules.
- In Flit Mode, the Downstream component must wait until it receives a Flit acknowledgement for the last Flit of the last TLP it had previously sent (i.e., the retry buffer is empty). The component must retransmit Flit(s) out of its Retry buffer if required by the Flit Ack/Nak rules.
- The Downstream component then initiates ASPM negotiation by sending a PM_Active_State_Request_L1 DLLP onto its Transmit Lanes. The Downstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Active_State_Request_L1 DLLP in Non-Flit Mode. The transmission of other DLLPs and SKP Ordered Sets must occur as required at any time between PM_Active_State_Request_L1

¹¹⁰ 110. Software must enable ASPM L1 for the Downstream component only if it is also enabled for the Upstream component.

transmissions, and do not contribute to this idle time limit. Transmission of SKP Ordered Sets during L1 entry follows the clock tolerance compensation rules in § [Section 4.2.8](#).

- The Downstream component continues to transmit the PM_Active_State_Request_L1 DLLP as described above until it receives a response from the Upstream device (see below). The Downstream component remains in this loop waiting for a response from the Upstream component.

During this waiting period, the Downstream component must not initiate any Transaction Layer transfers. It must still accept TLPs and DLLPs from the Upstream component, storing for later transmission any TLP responses required. It continues to respond with DLLPs, including FC update DLLPs, as needed by the Link Layer protocol.

If the Downstream component for any reason needs to transmit a TLP on the Link, it must first complete the transition to the low-power Link state. Once in a lower power Link state, the Downstream component must then initiate exit of the low-power Link state to handle the transfer. Refer to § [Section 5.2](#) if the negotiation to L1 is interrupted.

- The Upstream component must immediately (while obeying all other rules in this specification) respond to the request with either an acceptance or a rejection of the request.
If the Upstream component is not able to accept the request, it must immediately (while obeying all other rules in this specification) reject the request.
- Refer to § [Section 5.2](#) if the negotiation to L1 is interrupted.

Rules in case of rejection:

- In the case of a rejection, the Upstream component must schedule, as soon as possible, a rejection by sending the PM_Active_State_Nak Message to the Downstream component. Once the PM_Active_State_Nak Message is sent, the Upstream component is permitted to initiate any TLP or DLLP transfers.
- If the request was rejected, it is generally recommended that the Downstream component immediately transition its Transmit Lanes into the L0s state, provided L0s is enabled and that conditions for L0s entry are met.
- Prior to transmitting a PM_Active_State_Request_L1 DLLP associated with a subsequent ASPM L1 negotiation sequence, the Downstream component must either enter and exit L0s on its Transmitter, or it must wait at least 10 µs from the last transmission of the PM_Active_State_Request_L1 DLLP associated with the preceding ASPM L1 negotiation. This 10 µs timer must count only time spent in the LTSSM L0 and L0s states. The timer must hold in the LTSSM Recovery state. If the Link goes down and comes back up, the timer is ignored and the component is permitted to issue new ASPM L1 request after the Link has come back up.

IMPLEMENTATION NOTE: ASPM L1 ACCEPT/REJECT CONSIDERATIONS FOR THE UPSTREAM COMPONENT §

When the Upstream component has responded to the Downstream component's ASPM L1 request with a PM_Request_Ack DLLP to accept the L1 entry request, the ASPM L1 negotiation protocol clearly and unambiguously ends with the Link entering L1. However, if the Upstream component responds with a PM_Active_State_Nak Message to reject the L1 entry request, the termination of the ASPM L1 negotiation protocol is less clear. Therefore, both components need to be designed to unambiguously terminate the protocol exchange. If this is not done, there is the risk that the two components will get out of sync with each other, and the results may be undefined. For example, consider the following case:

- The Downstream component requests ASPM L1 entry by transmitting a sequence of PM_Active_State_Request_L1 DLLPs.
- Due to a temporary condition, the Upstream component responds with a PM_Active_State_Nak Message to reject the L1 request.
- The Downstream component continues to transmit the PM_Active_State_Request_L1 DLLPs for some time before it is able to respond to the PM_Active_State_Nak Message.
- Meanwhile, the temporary condition that previously caused the Upstream component to reject the L1 request is resolved, and the Upstream component erroneously sees the continuing PM_Active_State_Request_L1 DLLPs as a new request to enter L1, and responds by transmitting PM_Request_Ack DLLPs Downstream.

At this point, the result is undefined, because the Downstream component views the L1 request as rejected and finishing, but the Upstream component views the situation as a second L1 request being accepted.

To avoid this situation, the Downstream component needs to provide a mechanism to distinguish between one ASPM L1 request and another. The Downstream component does this by entering L0s (when supported and enabled), or by waiting a minimum of 10 µs from the transmission of the last PM_Active_State_Request_L1 DLLP associated with the first ASPM L1 request before starting transmission of the PM_Active_State_Request_L1 DLLPs associated with the second request (as described above).

If the Upstream component is capable of exhibiting the behavior described above, then it is necessary for the Upstream component to recognize the end of an L1 request sequence by detecting a transition to L0s on its Receiver (when supported and enabled) or a break in the reception of PM_Active_State_Request_L1 DLLPs of 9.5 µs measured while in L0 / L0s or more as a separation between ASPM L1 requests by the Downstream component.

If there is a possibility of ambiguity, the Upstream component should reject the L1 request to avoid potentially creating the ambiguous situation outlined above.

Rules in case of acceptance:

- If the Upstream component is ready to accept the request, it must block scheduling of any TLPs from the Transaction Layer.
- In Non-Flit Mode, the Upstream component then must wait until it receives a Data Link Layer acknowledgement for the last TLP it had previously sent. The Upstream component must retransmit a TLP if required by the Data Link Layer rules.

Base 6.4 vs Base 6.3

- In Flit Mode, the Upstream component then must wait until it receives a Data Link Layer acknowledgement for the last Flit of the last TLP it had previously sent. The Upstream component must retransmit Flit(s) if required by the Data Link Layer rules.
- Once all TLPs/Flits have been acknowledged, the Upstream component sends a PM_Request_Ack DLLP Downstream. The Upstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Request_Ack DLLP in Non-Flit Mode. The transmission of SKP Ordered Sets must occur as required at any time between PM_Request_Ack transmissions, and do not contribute to this idle time limit. Transmission of SKP Ordered Sets during L1 entry follows the clock tolerance compensation rules in § Section 4.2.8.
- The Upstream component continues to transmit the PM_Request_Ack DLLP as described above until it observes its Receive Lanes enter into the Electrical Idle state. Refer to § Chapter 4. for more details on the Physical Layer behavior.
- If the Upstream component needs, for any reason, to transmit a TLP on the Link after it sends a PM_Request_Ack DLLP, it must first complete the transition to the low-power state, and then initiate an exit from the low-power state to handle the transfer once the Link is back to L0. Refer to § Section 5.2 if the negotiation to L1 is interrupted.
 - The Upstream component must initiate an exit from L1 in this case even if it does not have the required flow control credit to transmit the TLP(s).
- When the Downstream component detects a PM_Request_Ack DLLP on its Receive Lanes (signaling that the Upstream device acknowledged the transition to L1 request), the Downstream component then ceases sending the PM_Active_State_Request_L1 DLLP, disables DLLP, TLP transmission and brings its Transmit Lanes into the Electrical Idle state.
- When the Upstream component detects an Electrical Idle on its Receive Lanes (signaling that the Downstream component has entered the L1 state), it then ceases to send the PM_Request_Ack DLLP, disables DLLP, TLP transmission and brings the Downstream direction of the Link into the Electrical Idle state.

Notes:

- The transaction Layer Completion Timeout mechanism is not affected by transition to the L1 state (i.e., it must keep counting).
- Flow Control Update timers are frozen while the Link is in L1 state to prevent a timer expiration that will unnecessarily transition the Link back to the L0 state.

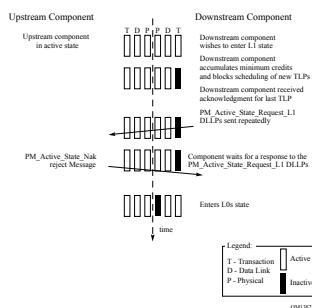


Figure 5-6 L1 Transition Sequence Ending with a Rejection (L0s Enabled) §

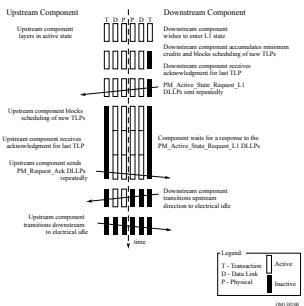


Figure 5-7 L1 Successful Transition Sequence §

5.4.1.3.2 Exit from the L1 State §

Components on either end of a Link may initiate an exit from the L1 Link state.

See § Section 5.5.1 for details on transitions into either the L1.1 or L1.2 substates.

Upon exit from L1, it is recommended that the Downstream component send flow control update DLLPs for all enabled VCs and FC types starting within 1 µs of L1 exit.

Downstream Component Initiated Exit

An Upstream Port must initiate an exit from L1 on its Transmit Lanes if it needs to communicate through the Link. The component initiates a transition to the L0 state as described in § Chapter 4.. The Upstream component must respond by initiating a similar transition of its Transmit Lanes.

If the Upstream component is a Switch Downstream Port, (i.e., it is not a Root Complex Root Port), the Switch must initiate an L1 exit transition on its Upstream Port's Transmit Lanes, (if the Upstream Port's Link is in the L1 state), as soon as it detects the L1 exit activity on any of its Downstream Port Links. Since L1 exit latencies are relatively long, a Switch must not wait until its Downstream Port Link has fully exited to L0 before initiating an L1 exit transition on its Upstream Port Link. Waiting until the Downstream Link has completed the L0 transition will cause a Message traveling through several Switches to experience accumulating latency as it traverses each Switch.

A Switch is required to initiate an L1 exit transition on its Upstream Port Link after no more than 1 µs from the beginning of an L1 exit transition on any of its Downstream Port Links. Refer to § Section 4.2 for details of the Physical Layer signaling during L1 exit.

Consider the example in § Figure 5-8 . The numbers attached to each Port represent the corresponding Port's reported Transmit Lanes L1 exit latency in units of microseconds.

Links 1, 2, and 3 are all in the L1 state, and Endpoint C initiates a transition to the L0 state at time T. Since Switch B takes 32 µs to exit L1 on its Ports, Link 3 will transition to the L0 state at T+32 (longest time considering T+8 for the Endpoint C, and T+32 for Switch B).

Switch B is required to initiate a transition from the L1 state on its Upstream Port Link (Link 2) after no more than 1 µs from the beginning of the transition from the L1 state on Link 3. Therefore, transition to the L0 state will begin on Link 2 at T+1. Similarly, Link 1 will start its transition to the L0 state at time T+2.

Following along as above, Link 2 will complete its transition to the L0 state at time T+33 (since Switch B takes longer to transition and it started at time T+1). Link 1 will complete its transition to the L0 state at time T+34 (since the Root Complex takes 32 µs to transition and it started at time T+2).

Therefore, among Links 1, 2, and 3, the Link to complete the transition to the L0 state last is Link 1 with a 34 µs delay. This is the delay experienced by the packet that initiated the transition in Endpoint C.

Base 6.4 vs Base 6.3

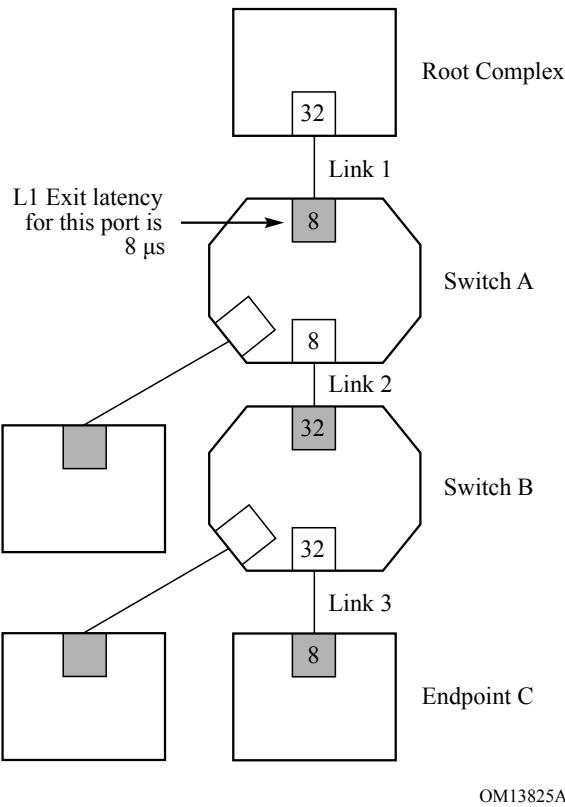


Figure 5-8 Example of L1 Exit Latency Computation

OMI3825A

Switches are not required to initiate an L1 exit transition on any other of their Downstream Port Links.

Upstream Component Initiated Exit

A Root Complex, or a Switch must initiate an exit from L1 on any of its Root Ports, or Downstream Port Links if it needs to communicate through that Link. The Switch or Root Complex must be capable of initiating L1 exit even if it does not have the flow control credits needed to transmit a given TLP. The component initiates a transition to the L0 state as described in § Chapter 4.. The Downstream component must respond by initiating a similar transition on its Transmit Lanes.

If the Downstream component contains a Switch, it must initiate a transition on all of its Downstream Links (assuming the Downstream Link is in an ASPM L1 state) as soon as it detects an exit from L1 state on its Upstream Port Link. Since L1 exit latencies are relatively long, a Switch must not wait until its Upstream Port Link has fully exited to L0 before initiating an L1 exit transition on its Downstream Port Links. If that were the case, a Message traveling through multiple Switches would experience accumulating latency as it traverses each Switch.

A Switch is required to initiate a transition from L1 state on all of its Downstream Port Links that are currently in L1 after no more than 1 μs from the beginning of a transition from L1 state on its Upstream Port. Refer to § Section 4.2 for details of the Physical Layer signaling during L1 exit. Downstream Port Links that are already in the L0 state do not participate in the exit transition. Downstream Port Links whose Downstream component is in a low power D-state (D1 - D3_{Hot}) are also not affected by the L1 exit transitions (i.e., such Links must not be transitioned to the L0 state).

5.4.1.4 ASPM Configuration §

All Functions must implement the following configuration bits in support of ASPM. Refer to § Chapter 7. for configuration register assignment and access mechanisms.

Each component reports its level of support for ASPM in the ASPM Support field below.

Table 5-3 Encoding of the ASPM Support Field §

| Field | Description |
|--------------|---------------------------------|
| ASPM Support | 00b No ASPM support |
| | 01b L0s supported |
| | 10b L1 supported |
| | 11b L0s and L1 supported |

Software must not enable L0s in either direction on a given Link unless components on both sides of the Link each support L0s ; otherwise, the result is undefined.

Each component reports the source of its reference clock in its Slot Clock Configuration bit located in its Capability structure's Link Status register.

Table 5-4 Description of the Slot Clock Configuration Bit §

| Bit | Description |
|--------------------------|--|
| Slot Clock Configuration | <p>This bit, when Set, indicates that the component uses the same physical reference clock that the platform provides on the connector.</p> <p>This bit, when Clear, indicates the component uses an independent clock irrespective of the presence of a reference on the connector.</p> <p>For Root and Switch Downstream Ports, this bit, when Set, indicates that the Downstream Port is using the same reference clock as the Downstream component or the slot.</p> <p>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.</p> <p>Otherwise it is Clear.</p> |

Each component must support the Common Clock Configuration bit in its Capability structure's Link Control register. Software writes to this register bit to indicate to the device whether it is sharing the same clock source as the device on the other end of the Link.

Table 5-5 Description of the Common Clock Configuration Bit §

| Bit | Description |
|----------------------------|--|
| Common Clock Configuration | This bit, when Set, indicates that this component and the component at the opposite end of the Link are operating with a common clock source. |
| | This bit, when Clear, indicates that this component and the component at the opposite end of the Link are operating with separate reference clock sources. |
| | Default value of this bit is 0b. |

| Bit | Description |
|-----|---|
| | Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies. |

Each Port reports the L0s and L1 exit latency (the time that they require to transition their Receive Lanes from the L0s or L1 state to the L0 state) in the L0s Exit Latency and the L1 Exit Latency configuration fields, respectively. If a Port does not support L0s or ASPM L1, the value of the respective exit latency field is undefined.

Table 5-6 Encoding of the L0s Exit Latency Field §

| Field | Description |
|--------------------|--|
| L0s Exit Latency > | 000b Less than 64 ns |
| | 001b 64 ns to less than 128 ns |
| | 010b 128 ns to less than 256 ns |
| | 011b 256 ns to less than 512 ns |
| | 100b 512 ns to less than 1 µs |
| | 101b 1 µs to less than 2 µs |
| | 110b 2 µs to 4 µs |
| | 111b More than 4 µs |

Table 5-7 Encoding of the L1 Exit Latency Field §

| Field | Description |
|-----------------|--------------------------------------|
| L1 Exit Latency | 000b Less than 1 µs |
| | 001b 1 µs to less than 2 µs |
| | 010b 2 µs to less than 4 µs |
| | 011b 4 µs to less than 8 µs |
| | 100b 8 µs to less than 16 µs |
| | 101b 16 µs to less than 32 µs |
| | 110b 32 µs to 64 µs |
| | 111b More than 64 µs |

Endpoints also report the additional latency that they can absorb due to the transition from L0s state or L1 state to the L0 state. This is reported in the Endpoint L0s Acceptable Latency and Endpoint L1 Acceptable Latency fields, respectively.

Power management software, using the latency information reported by all components in the Hierarchy, can enable the appropriate level of ASPM by comparing exit latency for each given path from Root to Endpoint against the acceptable latency that each corresponding Endpoint can withstand.

Base 6.4 vs Base 6.3

Table 5-8 Encoding of the Endpoint L0s Acceptable Latency Field §

| Field | Description |
|---------------------------------|-------------------------------|
| Endpoint L0s Acceptable Latency | 000b Maximum of 64 ns |
| | 001b Maximum of 128 ns |
| | 010b Maximum of 256 ns |
| | 011b Maximum of 512 ns |
| | 100b Maximum of 1 µs |
| | 101b Maximum of 2 µs |
| | 110b Maximum of 4 µs |
| | 111b No limit |

Table 5-9 Encoding of the Endpoint L1 Acceptable Latency Field §

| Field | Description |
|--------------------------------|------------------------------|
| Endpoint L1 Acceptable Latency | 000b Maximum of 1 µs |
| | 001b Maximum of 2 µs |
| | 010b Maximum of 4 µs |
| | 011b Maximum of 8 µs |
| | 100b Maximum of 16 µs |
| | 101b Maximum of 32 µs |
| | 110b Maximum of 64 µs |
| | 111b No limit |

Power management software enables or disables ASPM in each component by programming the ASPM Control field.

Table 5-10 Encoding of the ASPM Control Field §

| Field | Description |
|--------------|------------------------------|
| ASPM Control | 00b Disabled |
| | 01b L0s Entry Enabled |
| | 10b L1 Entry Enabled |

| Field | Description |
|------------|--------------------------|
| 11b | L0s and L1 Entry enabled |

ASPM Control = 00b

Port's Transmitter must not enter L0s .

Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

Ports connected to the Upstream end of the Link receiving a L1 request must respond with negative acknowledgement.

ASPM Control = 01b

Port must bring a Link into L0s state if all conditions are met.

Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

Ports connected to the Upstream end of the Link receiving a L1 request must respond with negative acknowledgement.

ASPM Control = 10b

Port's Transmitter must not enter L0s .

Ports connected to the Downstream end of the Link may issue PM_Active_State_Request_L1 DLLPs.

Ports connected to the Upstream end of the Link must respond with positive acknowledgement to a L1 request and transition into L1 if the conditions for the Root Complex Root Port or Switch Downstream Port in § Section 5.4.1.3.1 are met.

ASPM Control = 11b

Port must bring a Link into the L0s state if all conditions are met.

Ports connected to the Downstream end of the Link may issue PM_Active_State_Request_L1 DLLPs.

Ports connected to the Upstream end of the Link must respond with positive acknowledgement to a L1 request and transition into L1 if the conditions for the Root Complex Root Port or Switch Downstream Port in § Section 5.4.1.3.1 are met.

5.4.1.4.1 Software Flow for Enabling or Disabling ASPM §

Following is an example software algorithm that highlights how to enable or disable ASPM in a component.

- PCI Express components power up with an appropriate value in their Slot Clock Configuration bit. The method by which they initialize this bit is device-specific.
- PCI Express system software scans the Slot Clock Configuration bit in the components on both ends of each Link to determine if both are using the same reference clock source or reference clocks from separate sources. If the Slot Clock Configuration bits in both devices are Set, they are both using the same reference clock source, otherwise they're not.
- PCI Express software updates the Common Clock Configuration bits in the components on both ends of each Link to indicate if those devices share the same reference clock and triggers Link retraining by writing 1b to the Retrain Link bit in the Link Control register of the Upstream component.

- Devices must reflect the appropriate L0s / L1 exit latency in their L0s / L1 Exit Latency fields, per the setting of the Common Clock Configuration bit.
- PCI Express system software then reads and calculates the L0s / L1 exit latency for each Endpoint based on the latencies reported by each Port. Refer to § Section 5.4.1.3.2 for an example.
- For each component with one or more Endpoint Functions, PCI Express system software examines the Endpoint L0s Acceptable Latency / Endpoint L1 Acceptable Latency , as reported by each Endpoint Function in its Link Capabilities Register , and enables or disables L0s /L1 entry (via the ASPM Control field in the Link Control Register) accordingly in some or all of the intervening device Ports on that hierarchy.

5.5 L1 PM Substates §

L1 PM Substates establish a Link power management regime that creates lower power substates of the L1 Link state (see § Figure 5-9), and associated mechanisms for using those substates. The L1 PM Substates are:

- L1.0** substate
 - The L1.0 substate corresponds to the conventional L1 Link state. This substate is entered whenever the Link enters L1 . The L1 PM Substate mechanism defines transitions from this substate to and from the L1.1 and L1.2 substates.
 - The Upstream and Downstream Ports must be enabled to detect Electrical Idle exit as required in § Section 4.2.7.7.2 .
- L1.1** substate
 - Link common mode voltages are maintained.
 - Uses a bidirectional open-drain clock request (CLKREQ#) signal for entry to and exit from this state.
 - The Upstream and Downstream Ports are not required to be enabled to detect Electrical Idle exit.
- L1.2** substate
 - Link common mode voltages are not required to be maintained.
 - Uses a bidirectional open-drain clock request (CLKREQ#) signal for entry to and exit from this state.
 - The Upstream and Downstream Ports are not required to be enabled to detect Electrical Idle exit.

Ports that support L1 PM Substates must not require a reference clock while in L1 PM Substates other than L1.0 .

Ports that support L1 PM Substates and also support SRIS mode are required to support L1 PM Substates while operating in SRIS mode. In such cases the CLKREQ# signal is used by the L1 PM Substates protocol as defined in this section, but has no defined relationship to any local clocks used by either Port on the Link, and the management of such local clocks is implementation specific.

Ports that support the L1.2 substate for ASPM L1 must support Latency Tolerance Reporting (LTR).

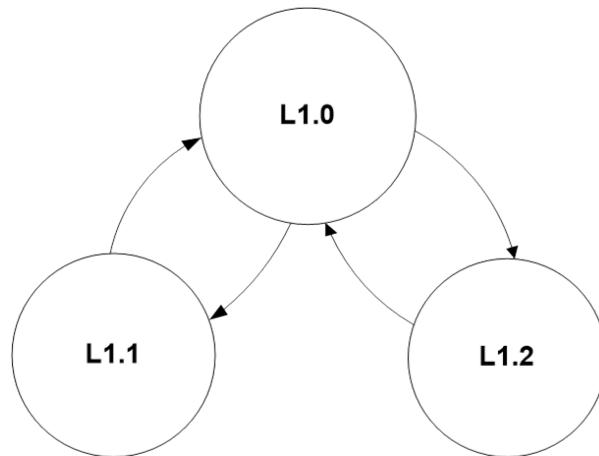


Figure 5-9 State Diagram for L1 PM Substates §

- When enabled, the L1 PM Substates mechanism applies the following additional requirements to the CLKREQ# signal: The CLKREQ# signal must be supported as a bi-directional open drain signal by both the Upstream and Downstream Ports of the Link. Each Port must have a unique instance of the signal, and the Upstream and Downstream Port CLKREQ# signals must be connected.
- It is permitted for the Upstream Port to deassert CLKREQ# when the Link is in the PCI-PM L1 or ASPM L1 states, or when the Link is in the L2/L3 Ready pseudo-state; CLKREQ# must be asserted by the Upstream Port when the Link is in any other state.
- All other specifications related to the CLKREQ# signal that are not specifically defined or modified by L1 PM Substates continue to apply.

If these requirements cannot be satisfied in a particular system, then L1 PM Substates must not be enabled.

IMPLEMENTATION NOTE: CLKREQ# CONNECTION TOPOLOGIES §

For an Upstream component the connection topologies for the CLKREQ# signal can vary. A few examples of CLKREQ# connection topologies are described below. For the Downstream component these cases are essentially the same, however from the Upstream component's perspective, there are some key differences that are described below.

Example 1: Single Downstream Port with a single PLL connected to a single Upstream Port (see § [Figure 5-10](#)).

In this platform configuration the Upstream component (A) has only a single CLKREQ# signal. The Upstream and Downstream Ports' CLKREQ# (A and B) signals are connected to each other. In this case, Upstream component (A), must assert CLKREQ# signal whenever it requires a reference clock.

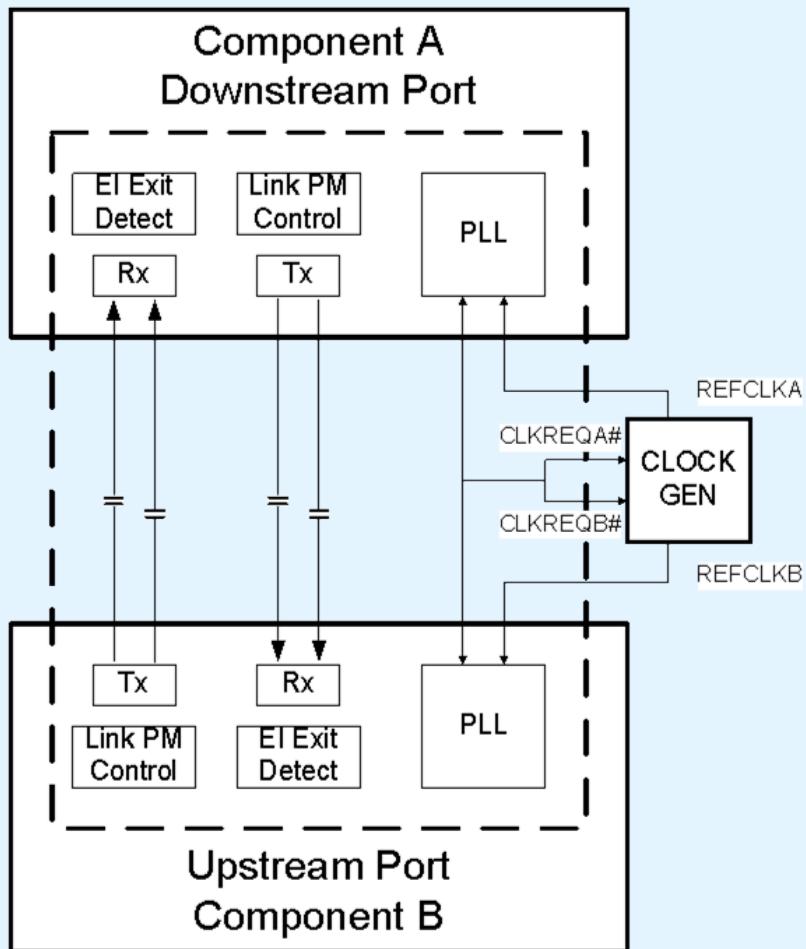


Figure 5-10 Downstream Port with a Single PLL §

Example 2: Upstream component with multiple Downstream Ports, with a common shared PLL, connected to separate Downstream components (see § [Figure 5-11](#)).

Base 6.4 vs Base 6.3

In this example configuration, there are three instances of CLKREQ# signal for the Upstream component (A), one per Downstream Port and a common shared CLKREQ# signal for the Upstream component (A). In this topology the Downstream Port CLKREQ# (CLKREQB#, CLKREQC#) signals are used to connect to the CLKREQ# signal of the Upstream Port of the Downstream components (B and C). The common shared CLKREQ# (CLKREQA#) signal for the Upstream component is used to request the reference clock for the shared PLL. The PLL control logic in Upstream component (A) can only be turned off and CLKREQA# be deasserted when both the Downstream Ports are in L1.1 or L1.2 Substates, and all internal (A) consumers of the PLL don't require a clock.

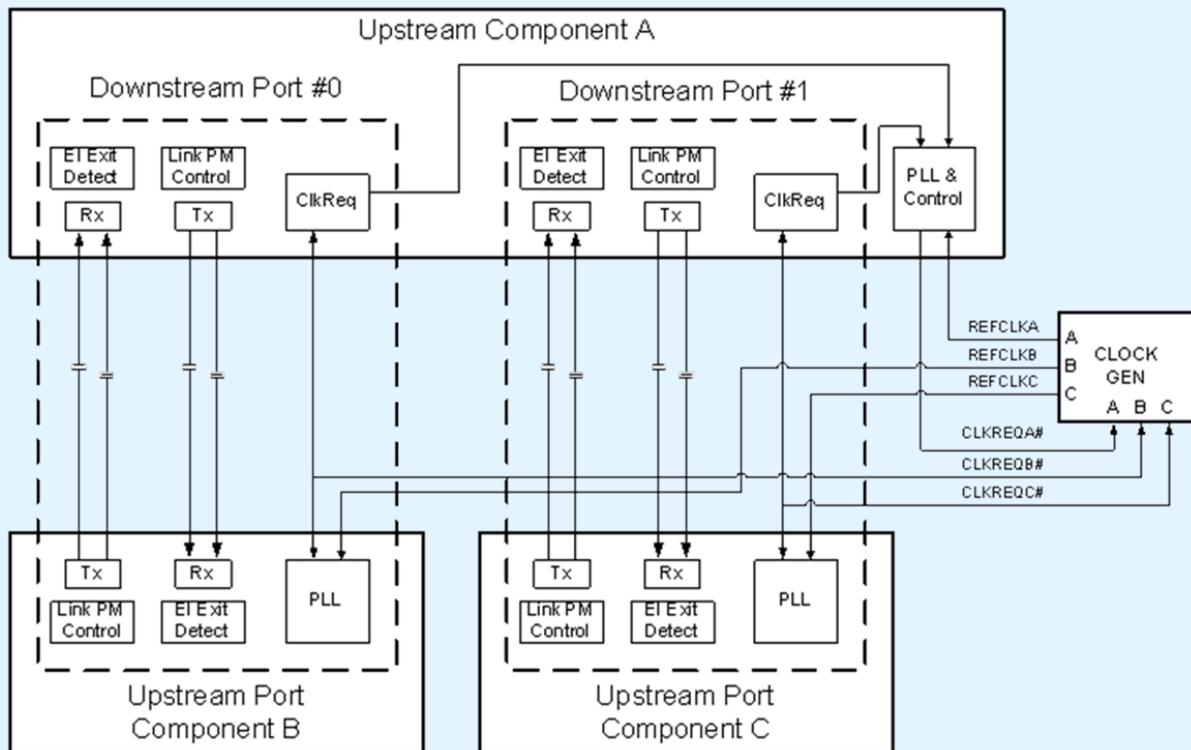


Figure 5-11 Multiple Downstream Ports with a shared PLL §

It is necessary for board implementers to consider what CLKREQ# topologies will be supported by components in order to make appropriate board level connections to support L1 PM Substates and for the reference clock generation.

IMPLEMENTATION NOTE: AVOIDING UNINTENDED INTERACTIONS BETWEEN L1 PM SUBSTATES AND THE LTSSM §

It is often the case that implementation techniques which save power will also increase the latency to return to normal operation. When implementing L1 PM Substates, it is important for the implementer to ensure that any added delays will not negatively interact with other elements of the platform. It is particularly important to ensure that LTSSM timeout conditions are not unintentionally triggered. Although typical implementations will not approach the latencies that would cause such interactions, the responsibility lies with the implementer to ensure that correct overall operation is achieved.

5.5.1 Entry conditions for L1 PM Substates and L1.0 Requirements §

The Link is considered to be in PCI-PM L1.0 when the L1 PM Substate is L1.0 and the LTSSM entered L1 through PCI-PM compatible power management. The Link is considered to be in ASPM L1.0 when the L1 PM Substate is in L1.0 and LTSSM entered L1 through ASPM.

The following rules define how the L1.1 and L1.2 substates are entered:

- Both the Upstream and Downstream Ports must monitor the logical state of the CLKREQ# signal.
- When in PCI-PM L1.0 and the PCI-PM L1.2 Enable bit is Set, the L1.2 substate must be entered when CLKREQ# is deasserted.
- When in PCI-PM L1.0 and the PCI-PM L1.1 Enable bit is Set, the L1.1 substate must be entered when CLKREQ# is deasserted and the PCI-PM L1.2 Enable bit is Clear.
- When in ASPM L1.0 and the ASPM L1.2 Enable bit is Set, the L1.2 substate must be entered when CLKREQ# is deasserted and all of the following conditions are true:
 - The reported snooped LTR value last sent or received by this Port is greater than or equal to the value set by the LTR_L1.2_THRESHOLD Value and Scale fields, or there is no snoop service latency requirement.
 - The reported non-snooped LTR last sent or received by this Port value is greater than or equal to the value set by the LTR_L1.2_THRESHOLD Value and Scale fields, or there is no non-snoop service latency requirement.
- When in ASPM L1.0 and the ASPM L1.1 Enable bit is Set, the L1.1 substate must be entered when CLKREQ# is deasserted and the conditions for entering the L1.2 substate are not satisfied.

When the entry conditions for L1.2 are satisfied, the following rules apply:

- Both the Upstream and Downstream Ports must monitor the logical state of the CLKREQ# input signal.
- An Upstream Port must not deassert CLKREQ# until the Link has entered L1.0.
- It is permitted for either Port to assert CLKREQ# to prevent the Link from entering L1.2.
- A Downstream Port intending to block entry into L1.2 must assert CLKREQ# before the Link enters L1.
- When CLKREQ# is deasserted the Ports enter the L1.2.Entry substate of L1.2.

If a Downstream Port is in PCI-PM L1.0 and PCI-PM L1.1 Enable and/or PCI-PM L1.2 Enable are Set, or if a Downstream Port is in ASPM L1.0 and ASPM L1.1 Enable and/or ASPM L1.2 Enable are Set, and the Downstream Port initiates an exit to Recovery without having entered L1.1 or L1.2 , the Downstream Port must assert CLKREQ# until the Link exits Recovery.

5.5.2 L1.1 Requirements §

Both Upstream and Downstream Ports are permitted to deactivate mechanisms for electrical idle (EI) exit detection and Refclk activity detection if implemented, however both ports must maintain common mode.

5.5.2.1 Exit from L1.1 §

If either the Upstream or Downstream Port needs to initiate exit from L1.1 , it must assert CLKREQ# until the Link exits Recovery. The Upstream Port must assert CLKREQ# on entry to Recovery, and must continue to assert CLKREQ# until the next entry into L1 , or other state allowing CLKREQ# deassertion.

- Next state is L1.0 if CLKREQ# is asserted.
 - The Refclk will eventually be turned on as defined in the PCI Express Mini CEM spec, which may be delayed according to the LTR advertised by the Upstream Port.

§ Figure 5-12 illustrates entry into L1.1 with exit driven by the Upstream Port.

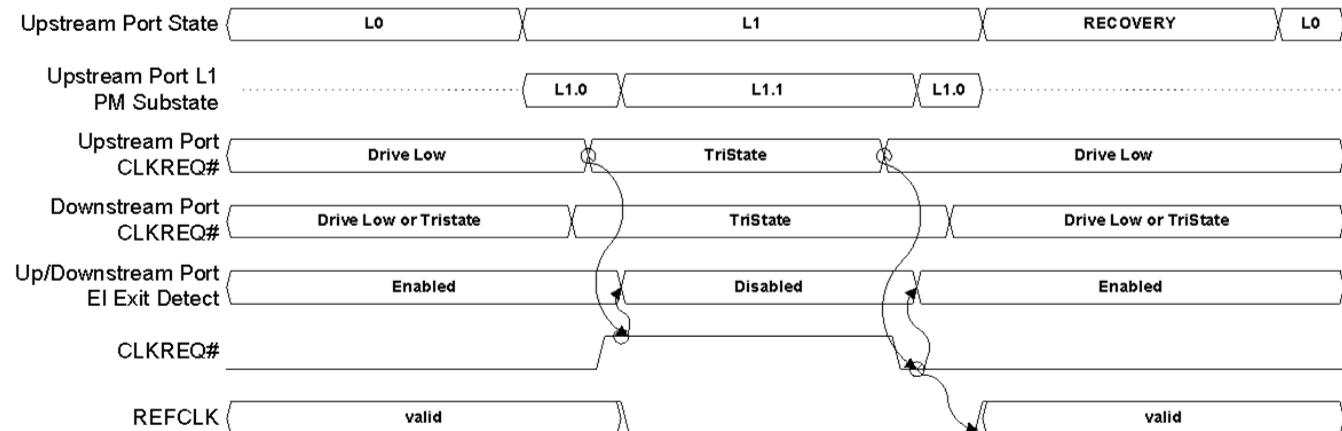


Figure 5-12 Example: L1.1 Waveforms Illustrating Upstream Port Initiated Exit §

§ Figure 5-13 illustrates entry into L1.1 with exit driven by the Downstream Port.

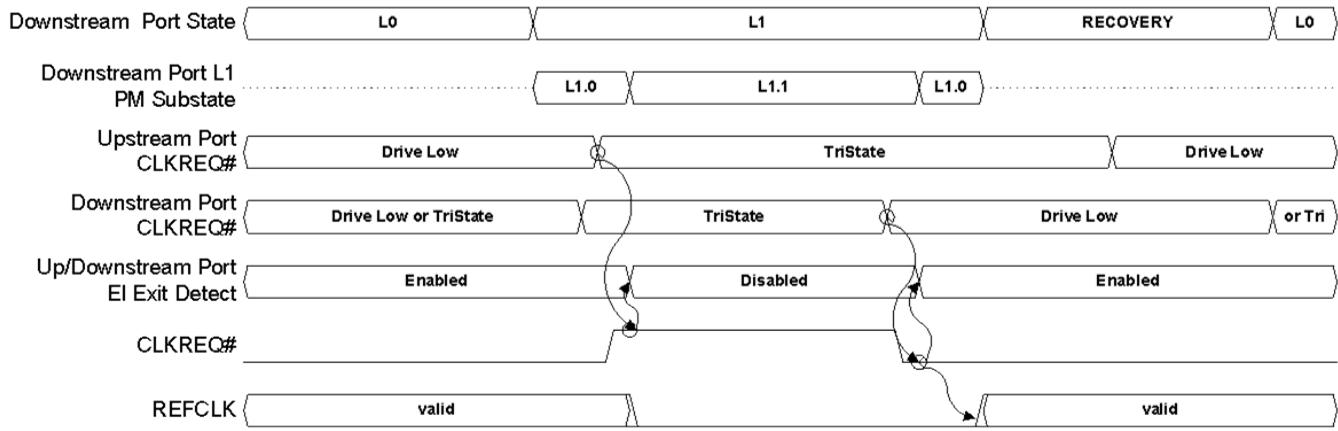


Figure 5-13 Example: L1.1 Waveforms Illustrating Downstream Port Initiated Exit §

5.5.3 L1.2 Requirements §

All Link and PHY state must be maintained during L1.2, or must be restored upon exit using implementation specific means, and the LTSSM and corresponding Port state upon exit from L1.2 must be indistinguishable from the L1.0 LTSSM and Port state.

L1.2 has additional requirements that do not apply to L1.1. These requirements are documented in this section.

L1.2 has three substates, which are defined below (see § Figure 5-14).

Base 6.4 vs Base 6.3

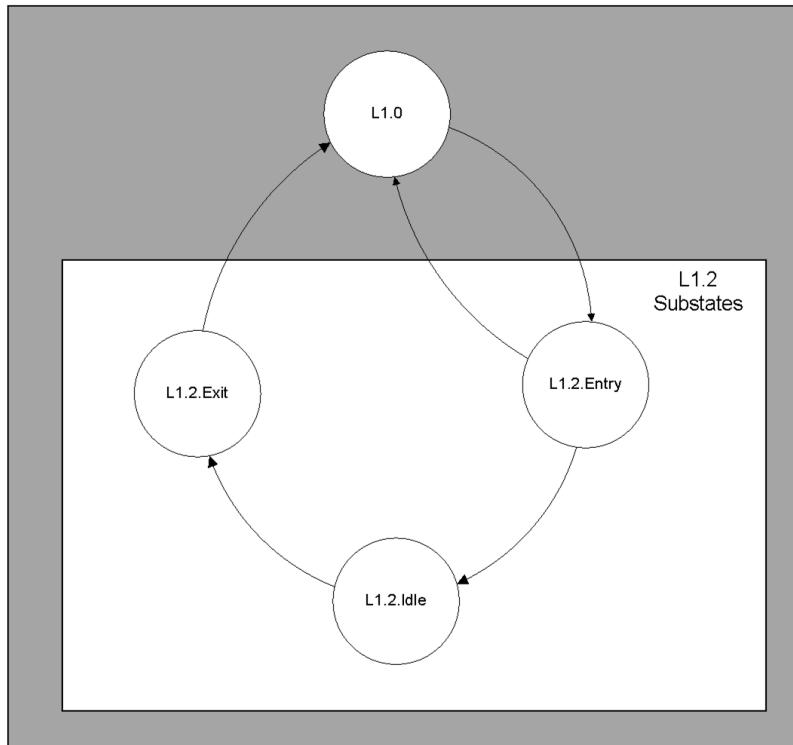


Figure 5-14 L1.2 Substates

5.5.3.1 L1.2.Entry

L1.2.Entry is a transitional state on entry into L1.2 to allow time for Refclk to turn off and to ensure both Ports have observed CLKREQ# deasserted. The following rules apply to L1.2.Entry:

- Both Upstream and Downstream Ports continue to maintain common mode.
- Both Upstream and Downstream Ports may turn off their electrical idle (EI) exit detect circuitry.
- The Upstream and Downstream Ports must not assert CLKREQ# in this state.
- Refclk must be turned off within $T_{L10_REFCLK_OFF}$.
- Next state is L1.0 if CLKREQ# is asserted, else the next state is L1.2.Idle after waiting for T_{POWER_OFF} .

Note that there is a boundary condition which can occur when one Port asserts CLKREQ# shortly after the other Port deasserts CLKREQ#, but before the first Port has observed CLKREQ# deasserted. This is an unavoidable boundary condition that implementations must handle correctly. An example of this condition is illustrated in § Figure 5-15 .

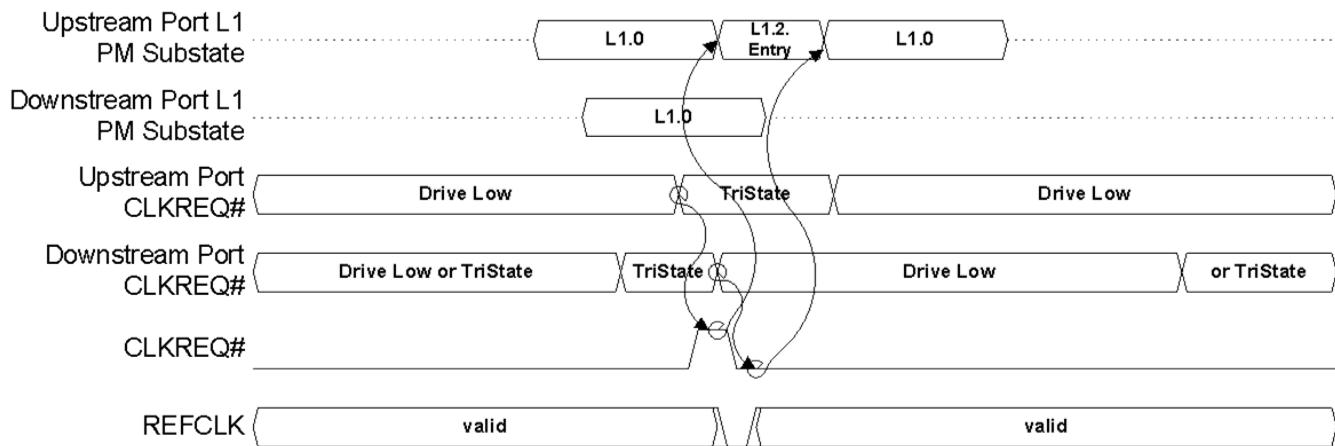


Figure 5-15 Example: Illustration of Boundary Condition due to Different Sampling of CLKREQ# §

5.5.3.2 L1.2.Idle §

When requirements for the entry into L1.2.Idle state (see § Section 5.5.1) have been satisfied then the Ports enter the L1.2.Idle substate. The following rules apply in L1.2.Idle :

- Both Upstream and Downstream Ports may power-down any active logic, including circuits required to maintain common mode.
- The PHY of both Upstream and Downstream Ports may have their power removed.

The following rules apply for L1.2.Idle state when using the CLKREQ#-based mechanism:

- If either the Upstream or Downstream Port needs to exit L1.2, it must assert CLKREQ# after ensuring that T_{L1.2} has been met.
- If the Downstream Port is initiating exit from L1, it must assert CLKREQ# until the Link exits Recovery. The Upstream Port must assert CLKREQ# on entry to Recovery, and must continue to assert CLKREQ# until the next entry into L1, or other state allowing CLKREQ# deassertion.
- If the Upstream Port is initiating exit from L1, it must continue to assert CLKREQ# until the next entry into L1, or other state allowing CLKREQ# deassertion.
- Both the Upstream and Downstream Ports must monitor the logical state of the CLKREQ# input signal.
- Next state is L1.2.Exit if CLKREQ# is asserted.

5.5.3.3 L1.2.Exit §

This is a transitional state on exit from L1.2 to allow time for both devices to power up. In L1.2.Exit, the following rules apply:

- The PHYs of both Upstream and Downstream Ports must be powered.
- It must not be assumed that common mode has been maintained.

5.5.3.3.1 Exit from L1.2 §

- The following rules apply for L1.2.Exit using the CLKREQ#-based mechanism:
- Both Upstream and Downstream Ports must power up any circuits required for L1.0, including circuits required to maintain common mode.
- The Upstream and Downstream Ports must not change their driving state of CLKREQ# in this state.
- Refclk must be turned on no earlier than T_{L10_REFCLK_ON} minimum time, and may take up to the amount of time allowed according to the LTR advertised by the Endpoint before becoming valid.
- Next state is L1.0 after waiting for T_{POWER_ON}.
 - Common mode is permitted to be established passively during L1.0, and actively during Recovery. In order to ensure common mode has been established, the Downstream Port must maintain a timer, and the Downstream Port must continue to send TS1 training sequences until a minimum of T_{COMMONMODE} has elapsed since the Downstream Port has started transmitting TS1 training sequences and has detected electrical idle exit on any Lane of the configured Link.

§ Figure 5-16 illustrates the signal relationships and timing constraints associated with L1.2 entry and Upstream Port initiated exit.

§ Figure 5-17 illustrates the signal relationships and timing constraints associated with L1.2 entry and Downstream Port initiated exit.

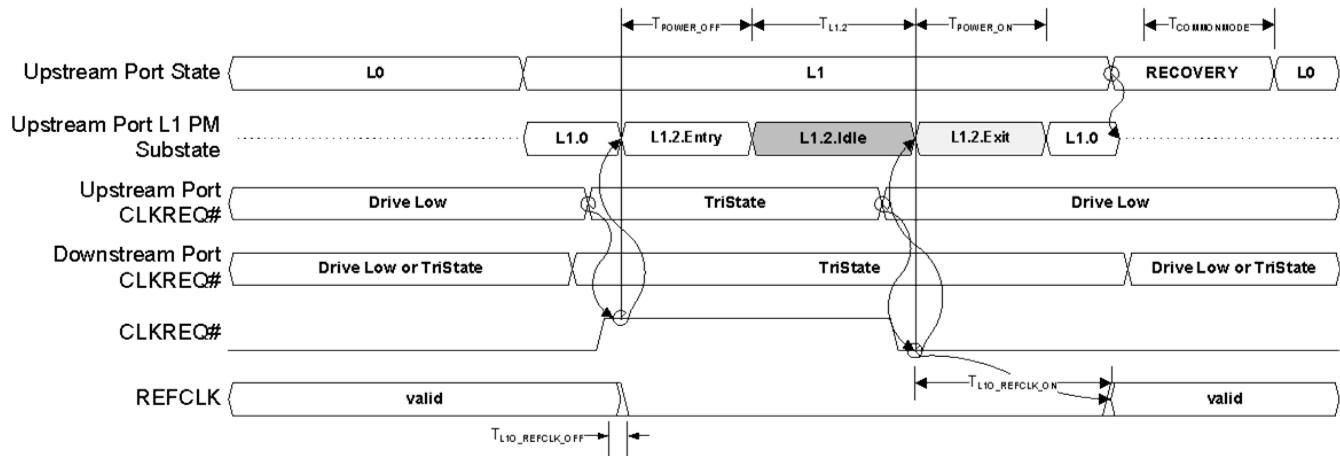


Figure 5-16 Example: L1.2 Waveforms Illustrating Upstream Port Initiated Exit §

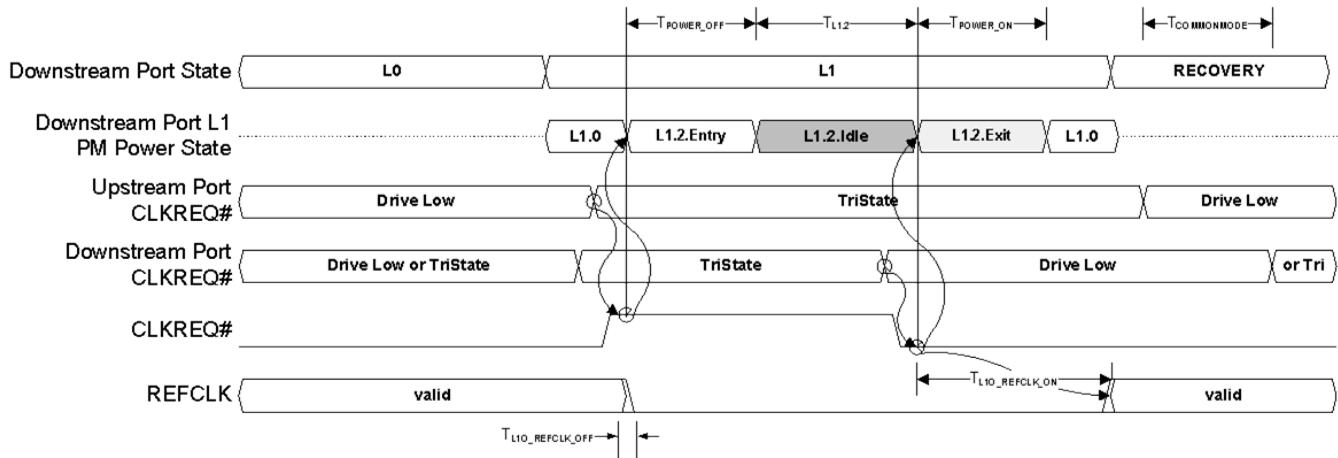


Figure 5-17 Example: L1.2 Waveforms Illustrating Downstream Port Initiated Exit §

5.5.4 L1 PM Substates Configuration §

L1 PM Substates is considered enabled on a Port when any combination of the ASPM L1.1 Enable, ASPM L1.2 Enable, PCI-PM L1.1 Enable and PCI-PM L1.2 Enable bits associated with that Port are Set.

An L1 PM Substate enable bit must only be Set in the Upstream and Downstream Ports on a Link when the corresponding supported capability bit is Set by both the Upstream and Downstream Ports on that Link, otherwise the behavior is undefined.

The Setting of any enable bit must be performed at the Downstream Port before the corresponding bit is permitted to be Set at the Upstream Port. If any L1 PM Substates enable bit is at a later time to be cleared, the enable bit(s) must be cleared in the Upstream Port before the corresponding enable bit(s) are permitted to be cleared in the Downstream Port.

If setting either or both of the enable bits for ASPM L1 PM Substates, both ports must be configured as described in this section while ASPM L1 is disabled.

If setting either or both of the enable bits for PCI-PM L1 PM Substates, both ports must be configured as described in this section while in D0.

Prior to setting either or both of the enable bits for L1.2, the values for T_{POWER_ON}, Common_Mode_Restore_Time, and, if the ASPM L1.2 Enable bit is to be Set, the LTR_L1.2_THRESHOLD (both Value and Scale fields) must be programmed.

The T_{POWER_ON} and Common_Mode_Restore_Time fields must be programmed to the appropriate values based on the components and AC coupling capacitors used in the connection linking the two components. The determination of these values is design implementation specific.

When both the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are cleared, it is not required to program the T_{POWER_ON}, Common_Mode_Restore_Time, and LTR_L1.2_THRESHOLD Value and Scale fields, and hardware must not rely on these fields to have any particular values.

When programming LTR_L1.2_THRESHOLD Value and Scale fields, identical values must be programmed in both Ports.

5.5.5 L1 PM Substates Timing Parameters §

§ Table 5-11 defines the timing parameters associated with the L1.2 substates mechanism.

Table 5-11 L1.2 Timing Parameters §

| Parameter | Description | Min | Max | Units |
|------------------------|---|--|--------------------------------------|-------|
| T_{POWER_OFF} | CLKREQ# deassertion to entry into the L1.2.Idle substate | | 2 | μs |
| $T_{COMMONMODE}$ | Restoration of Refclk to restoration of common mode established through active transmission of TS1 training sequences (see § Section 5.5.3.3.1) | Programmable in range from 0 to 255 | | μs |
| $T_{L10_REFCLK_OFF}$ | CLKREQ# deassertion to Refclk reaching idle electrical state when entering L1.2 | 0 | 100 | ns |
| $T_{L10_REFCLK_ON}$ | CLKREQ# assertion to Refclk valid when exiting L1.2 | T_{POWER_ON} | LTR value advertised by the Endpoint | μs |
| T_{POWER_ON} | The minimum amount of time that each component must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface to ensure no device is ever actively driving into an unpowered component. | Set in the L1 PM Substates Control 2 Register (range from 0 to 3100) | | μs |
| $T_{L1.2}$ | Time a Port must stay in L1.2 when CLKREQ# must remain inactive | 4 | | μs |

5.5.6 Link Activation §

Link Activation is an optional mechanism to temporarily disable L1 Substates. Link Activation is used to bring a Link out of L1.1 / L1.2, avoiding potential stalls. An example of one such stall is the stall associated with a Configuration Write to perform a D3Hot to D0 transition. Link Activation can also be used to indirectly indicate to a Device that it should avoid long-latency internal power management during latency-sensitive or time critical operations.

The following rules apply to Link Activation :

- A Downstream Port is permitted to support Link Activation , as indicated by the Link Activation Supported bit in the L1 PM Substates Capabilities Register being Set.
- The Link Activation Control bit must have no effect on Port behavior unless one or more of the following bits are Set:
 - PCI-PM L1.2 Enable
 - PCI-PM L1.1 Enable
- When the Link Activation Control bit is Set, the Port that is about to enter L1 must assert, and while in L1 maintain as asserted, the CLKREQ# signal.
- If the Link Activation Control bit is Clear, the Link Activation mechanism does not impose any additional requirements on the state of the CLKREQ# signal.
- If the Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:
 - The associated vector is unmasked (not applicable if MSI does not support PVM)
 - The Link Activation Interrupt Enable bit is Set
 - The Link Activation Control bit is Set
 - The Link Activation Status bit is Set. Note that Link Activation interrupts always use the MSI or MSI-X vector indicated by the Interrupt Message Number field in the PCI Express Capabilities Register .

- If the Port is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as the following conditions are satisfied:
 - The Interrupt Disable bit in the Command Register is Clear.
 - The Link Activation Interrupt Enable bit is Set
 - The Link Activation Control bit is Set
 - The Link Activation Status bit is Set
- The Link Activation Status bit must be Set every time the logical AND of the following conditions transitions from FALSE to TRUE:
 - Either the PCI-PM L1.2 Enable bit or the PCI-PM L1.1 Enable bit (or both) are Set
 - The Link Activation Control bit is Set
 - The Link is not in an L1 Substate

5.6 Auxiliary Power Support §

The specific definition and requirements associated with auxiliary power are form factor specific, and the terms “auxiliary power” and “Vaux” should be understood in reference to the specific form factor in use. The specific mechanism(s) for supplying auxiliary power are not defined in this specification. The following text defines requirements that apply in all form factors.

Note that support for auxiliary power is optional. Some form factors do not support it. Also, some form factors have dedicated auxiliary power pins while other form factors use the main power pins in some fashion.

PCI Express PM provides a Aux Power PM Enable bit in the Device Control Register that provides the means for enabling a Function to draw the maximum allowance of auxiliary current independent of its level of support for PME generation.

A Function requests auxiliary power allocation by specifying a non-zero value in the Aux_Current field of the PMC register. Refer to § Chapter 7. for the Aux Power PM Enable register bit assignment, and access mechanism.

Allocation of auxiliary power using Aux Power PM Enable and PME_En is determined as follows:

Table 5-12 Aux Power Source and Availability §

| Aux Power PM Enable | PME_En | Aux Power Detected | Aux Power Source | Aux Power Available |
|---------------------|--------|--------------------|--|---|
| x | x | 0b | None | None |
| 0b | 0b | 1b | Form factor specific Aux Power rail / pins | Form factor specific (e.g., 10 mW in CEM) |
| | 1b | | Form factor specific Aux Power rail / pins | Aux_Current value (PMC) |
| 1b | x | | Form factor specific Aux Power rail / pins | Aux_Current value (PMC) |

Aux Power PM Enable = 1b:

Auxiliary power is allocated as requested in the Aux_Current field of the PMC register, independent of the PME_En bit in the PMSCR. The PME_En bit still controls the ability to master PME.

Additional Aux power is permitted to be allocated using a firmware based mechanism (see the Request D3 Cold Aux Power Limit _DSM call as defined in [Firmware]).

Additional Aux power is also permitted to be allocated by selecting a PM Sub State in the Power Limit mechanism (see § Section 7.8.1.3).

Aux Power PM Enable = 0b:

Auxiliary power allocation is controlled by the PME_En bit as defined in § Section 7.5.2.2 .

Additional Aux power is permitted to be allocated using a firmware based mechanism (see the Request D3 Cold Aux Power Limit _DSM call as defined in [Firmware]).

Additional Aux power is also permitted to be allocated by selecting a PM Sub State in the Power Limit mechanism (see § Section 7.8.1.3).

The Aux Power PM Enable bit is sticky (see § Section 7.4) so its state is preserved in the D3Cold state, and is not affected by the transitions from the D3Cold state to the D0uninitialized state.

5.7 Power Management System Messages and DLLPs §

§ Table 5-13 defines the location of each PM packet in the PCI Express stack.

Table 5-13 Power Management System Messages and DLLPs §

| Packet | Type |
|----------------------------|---------------------------|
| PM_Enter_L1 | DLLP |
| PM_Enter_L23 | DLLP |
| PM_Active_State_Request_L1 | DLLP |
| PM_Request_Ack | DLLP |
| PM_Active_State_Nak | Transaction Layer Message |
| PM_PME | Transaction Layer Message |
| PME_Turn_Off | Transaction Layer Message |
| PME_TO_Ack | Transaction Layer Message |

For information on the structure of the power management DLLPs, refer to § Section 3.5 .

Power Management Messages follow the general rules for all Messages. Power Management Message fields follow the following rules:

- Length field is Reserved.
- Attribute field must be set to the default values (all 0's).
- Address field is Reserved.
- Requester ID - see § Table 2-23 in § Section 2.2.8.2 .
- Traffic Class field must use the default class (TC0).

5.8 PCI Function Power State Transitions §

All PCI-PM power management state changes are explicitly controlled by software except for Fundamental Reset which brings all Functions to the D0 uninitialized state. § Figure 5-18 shows all supported state transitions. The unlabeled arcs represent a software initiated state transition (Set Power State operation).

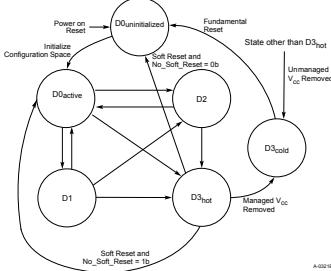


Figure 5-18 Function Power Management State Transitions §

5.9 State Transition Recovery Time Requirements §

§ Table 5-14 shows the minimum recovery times that system software must allow between the time that a Function is programmed to change state and the time that the function is next accessed (including Configuration Space), unless Readiness Notifications (see § Section 6.22) is used to indicate modified values to system software. For bridge Functions, this delay also constitutes a minimum delay between when the bridge's state is changed and when any Function on the logical bus that it originates can be accessed.

Table 5-14 PCI Function State Transition Delays §

| Initial State | Next State | Minimum System Software Guaranteed Delays |
|----------------------|---------------|---|
| <u>D0</u> | <u>D1</u> | 0 |
| <u>D0 or D1</u> | <u>D2</u> | 200 µs |
| <u>D0 , D1 or D2</u> | <u>D3 Hot</u> | 10 ms |
| <u>D1</u> | <u>D0</u> | 0 |
| <u>D2</u> | <u>D0</u> | 200 µs |
| <u>D3 Hot</u> | <u>D0</u> | 10 ms |

5.10 SR-IOV Power Management §

This section defines power management requirements that are unique to ↑↓SR-IOV devices.↓ ↑↑SR-IOV Devices .↑

The PCI Power Management Capability as described elsewhere in § Chapter 5. is required for PFs.

For VFs, the PCI Power Management Capability is optional.

5.10.1 VF Device Power Management States §

If a VF does not implement the PCI Power Management Capability, then the VF behaves as if it had been programmed into the equivalent power state of its associated PF.

If a VF implements the PCI Power Management Capability, the functionality must be as defined in § Section 7.5.2.

If a VF implements the PCI Power Management Capability, the Device behavior is undefined if the PF is placed in a lower power state than the VF. Software should avoid this situation by placing all VFs in lower power state before lowering their associated PF's power state.

A VF in the D0 state is in the D0_{active} state when the VF has completed its internal initialization and either the VF's Bus Master Enable bit or the VF MSE bit in the SR-IOV Control Register (see § Section 9.4.3.3) Extended Capability is Set. The VF's internal initialization must have completed when any of the following conditions have occurred:

- The VF has responded successfully (without returning RRS) to a Configuration Request.
- After issuing an FLR to the VF, one of the following is true:
 - At least 1.0 s has passed since the FLR was issued.
 - The VF supports Function Readiness Status and, after the FLR was issued, an FRS Message from the VF with Reason Code FLR Completed has been received.
 - At least FLR time has passed since the FLR was issued. FLR Time is either (1) the FLR Time value in the Readiness Time Reporting Extended Capability associated with the VF or (2) a value determined by system software / firmware ¹¹¹.
- After VF Enable has been Set in a PF, at least one of the following is true:
 - At least 1.0 s has passed since VF Enable was Set.
 - The PF supports Function Readiness Status and, after VF Enable was Set, an FRS Message from the PF with Reason Code VF Enabled has been received.
 - The Reset Time period in the VF's Readiness Time Reporting Extended Capability has passed.
 - A time period determined by platform firmware has passed.
 - A time period determined by non-guest OS software has passed.
- After transitioning a VF from D3_{Hot} to D0, at least one of the following is true:
 - At least 10 ms has passed since the request to enter D0 was issued.
 - The VF supports Function Readiness Status and, after the request to enter D0 was issued, an FRS Message from the VF with Reason Code D3_{Hot} to D0 Transition Completed has been received.
 - At least D3_{Hot} to D0 Time has passed since the request to enter D0 was issued. D3_{Hot} to D0 Time is either (1) the D3_{Hot} to D0 Time in the Readiness Time Reporting Extended Capability associated with the VF or (2) a value determined by system software / firmware ¹¹².

5.10.2 PF Device Power Management States §

The PF's power management state (D-state) has global impact on its associated VFs. If a VF does not implement the PCI Power Management Capability, then it behaves as if it is in an equivalent power state of its associated PF.

¹¹¹. For example, ACPI tables.

¹¹². For example, ACPI tables.

If a VF implements the PCI Power Management Capability, the Device behavior is undefined if the PF is placed in a lower power state than the VF. Software should avoid this situation by placing all VFs in lower power state before lowering their associated PF's power state.

When the PF is placed into the D3Hot state:

- If the No_Soft_Reset bit is Clear then the PF performs an internal reset on the D3Hot to D0 transition and all its configuration state returns to the default values.

Note: Resetting the PF resets VF Enable which means that VFs no longer exist and any VF specific context is lost after the D3Hot to D0 transition is complete.

- If the No_Soft_Reset bit is Set then the internal reset does not occur. The SR-IOV extended capability retains state, and associated VFs remain enabled.

When the PF is placed into the D3Cold state VFs no longer exist, any VF specific context is lost and PME events can only be initiated by the PF.

IMPLEMENTATION NOTE: NO_SOFT_RESET STRONGLY RECOMMENDED §

It is strongly recommended that the No_Soft_Reset bit be Set in all Functions of a Multi-Function Device. As indicated in the bit definition, all implementations that support Flit Mode are required to Set the No_Soft_Reset bit. This recommendation applies to PFs.

5.11 PCI Bridges and Power Management §

With power management under the direction of the operating system, each class of Functions must have a clearly defined criteria for feature availability as well as what functional context must be preserved when operating in each of the power management states. Some example Device-Class specifications have been proposed as part of the ACPI specification for various Functions ranging from audio to network add-in cards. While defining Device-Class specific behavioral policies for most Functions is outside the scope of this specification, defining the required behavior for PCI bridge functions is within the scope of this specification. The definitions here apply to all three types of PCIe Bridges:

- Host bridge, PCI Express to expansion bus bridge, or other ACPI enumerated bridge
- Switches
- PCI Express to PCI bridge
- PCI-to-CardBus bridge

The mechanisms for controlling the state of these Functions vary somewhat depending on which type of Originating Device is present. The following sections describe how these mechanisms work for the three types of bridges.

This section details the power management policies for PCI Express Bridge Functions. The PCI Express Bridge Function can be characterized as an Originating Device with a secondary bus downstream of it. This section describes the relationship of the bridge function's power management state to that of its secondary bus.

The shaded regions in § Figure 5-19 illustrate what is discussed in this section.

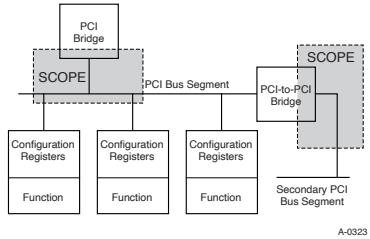


Figure 5-19 PCI Express Bridge Power Management Diagram §

As can be seen from § Figure 5-19 , the PCI Express Bridge behavior described in this chapter is common, from the perspective of the operating system, to host bridges, Switches, and PCI Express to PCI bridges.

It is the responsibility of the system software to ensure that only valid, workable combinations of bus and downstream Function power management states are used for a given bus and all Functions residing on that bus.

5.11.1 Switches and PCI Express to PCI Bridges §

The power management policies for the secondary bus of a Switch or PCI Express to PCI bridge are identical to those defined for any Bridge Function.

The BPCC_En and B2_B3# bus power/clock control fields in the Bridge Function's PMCSR_BSE register support the same functionality as for any other Bridges.

5.12 Power Management Events §

There are two varieties of Power Management Events:

- Wakeup Events
- PME Generation

A Wakeup Event is used to request that power be turned on.

A PME Generation Event is used to identify to the system the Function requesting that power be turned on.

In conventional PCI, both events are associated with the PME# signal. The PME# signal is asserted by a Function to request a change in its power management state. When the PME_En bit is Set and the event occurs, the Function sets the PME_Status bit and asserts the PME# signal. It keeps the PME# signal asserted until either the PME_En bit or the PME_Status are Cleared (typically by software).

In PCI Express, the Wakeup Event is associated with the WAKE# signal. If supported, the WAKE# signal is defined in the associated form factor specification and is used by a Function to request a change in its PCI-PM power management state when the Function is in D3Cold and PME_En is Set.

In PCI Express, after main power has been restored and the Link is trained, the Function(s) that initiated the wakeup (e.g., that asserted WAKE#), sends a PM_PME Message to the Root Complex. The PM_PME Message provides the Root Complex with the identity of the requesting Function(s) without requiring software to poll for the PME_Status bit being Set.

Base 6.4 vs Base 6.3

6. System Architecture §

This chapter addresses various aspects of PCI Express interconnect architecture in a platform context.

6.1 Interrupt and PME Support §

The PCI Express interrupt model supports two mechanisms:

- INTx emulation
- Message Signaled Interrupt (MSI/MSI-X)

For legacy compatibility, PCI Express provides a PCI INTx emulation mechanism to signal interrupts to the system interrupt controller (typically part of the Root Complex). This mechanism is compatible with existing PCI software, and provides the same level and type of service as the corresponding PCI interrupt signaling mechanism and is independent of system interrupt controller specifics. This legacy compatibility mechanism allows boot device support without requiring complex BIOS-level interrupt configuration/control service stacks. It virtualizes PCI physical interrupt signals by using an in-band signaling mechanism.

If an Endpoint Function supports interrupts, then this specification requires support of either MSI or MSI-X or both. PCI Compatible INTx interrupt emulation is optional. Switches are required to support forwarding the INTx interrupt emulation Messages (see § [Section 2.2.8.1](#)). The PCI Express MSI and MSI-X mechanisms are compatible with those originally defined in [PCI].

For ~~↓↑SR-IOV devices, ↓↑~~ ↑↑SR-IOV Devices, ↑ PFs are permitted to implement INTx, and VFs must not implement INTx. Each PF and VF must implement its own unique interrupt capabilities.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

6.1.1 Rationale for PCI Express Interrupt Model §

PCI Express takes an evolutionary approach from PCI with respect to interrupt support.

As required for PCI/PCI-X interrupt mechanisms, each device Function is required to differentiate between INTx and MSI/MSI-X modes of operation. The device complexity required to support both schemes is no different than that for PCI/PCI-X devices. The advantages of this approach include:

- Compatibility with existing PCI Software Models
- Direct support for boot devices
- Easier End of Life (EOL) for INTx legacy mechanisms.

The existing software model is used to differentiate INTx vs. MSI/MSI-X modes of operation; thus, no special software support is required for PCI Express.

The software model does not support changing interrupt modes while the Function is in active operation. If software does this, interrupt conditions may be dropped or replicated.

6.1.2 PCI-compatible INTx Emulation §

PCI Express emulates the PCI interrupt mechanism including the Interrupt Pin and Interrupt Line registers of the PCI Configuration Space for PCI device Functions. PCI Express non-Switch devices may optionally support these registers for backwards compatibility. Switch devices are required to support them. Actual interrupt signaling uses in-band Messages rather than being signaled using physical pins.

Two types of Messages are defined, Assert_INTx and Deassert_INTx, for emulation of PCI INTx signaling, where x is A, B, C, and D for respective PCI interrupt signals. These Messages are used to provide “virtual wires” for signaling interrupts across a Link. Switches collect these virtual wires and present a combined set at the Switch’s Upstream Port. Ultimately, the virtual wires are routed to the Root Complex which maps the virtual wires to system interrupt resources. Devices must use assert/deassert Messages in pairs to emulate PCI interrupt level-triggered signaling. Actual mapping of PCI Express INTx emulation to system interrupts is implementation specific as is mapping of physical interrupt signals in conventional PCI.

The legacy INTx emulation mechanism may be deprecated in a future version of this specification.

6.1.3 INTx Emulation Software Model §

The software model for legacy INTx emulation matches that of PCI. The system BIOS reporting of chipset/platform interrupt mapping and the association of each device Function’s interrupt with PCI interrupt lines is handled in exactly the same manner as with conventional PCI systems. Legacy software reads from each device Function’s Interrupt Pin register to determine if the Function is interrupt driven. A value between 01h and 04h indicates that the Function uses an emulated interrupt pin to generate an interrupt.

Note that similarly to physical interrupt signals, the INTx emulation mechanism may potentially cause spurious interrupts that must be handled by the system software.

6.1.4 MSI and MSI-X Operation §

Message Signaled Interrupts (MSI) is an optional feature that enables a device Function to request service by writing a system-specified data value to a system-specified address (using a DWORD Memory Write transaction). System software initializes the message address and message data (from here on referred to as the “vector”) during device configuration, allocating one or more vectors to each MSI-capable Function.

Interrupt latency (the time from interrupt signaling to interrupt servicing) is system dependent. Consistent with current interrupt architectures, Message Signaled Interrupts do not provide interrupt latency time guarantees.

MSI-X defines a separate optional extension to basic MSI functionality. Compared to MSI, MSI-X supports a larger maximum number of vectors per Function, the ability for software to control aliasing when fewer vectors are allocated than requested, plus the ability for each vector to use an independent address and data value, specified by a table that resides in Memory Space. However, most of the other characteristics of MSI-X are identical to those of MSI.

For the sake of software backward compatibility, MSI and MSI-X use separate and independent Capability structures. On Functions that support both MSI and MSI-X, system software that supports only MSI can still enable and use MSI without any modification. MSI functionality is managed exclusively through the MSI Capability structure, and MSI-X functionality is managed exclusively through the MSI-X Capability structure.

A Function is permitted to implement both MSI and MSI-X, but system software is prohibited from enabling both at the same time. If system software enables both at the same time, the behavior is undefined.

All PCI Express Endpoint Functions that are capable of generating interrupts must support MSI or MSI-X or both. The MSI and MSI-X mechanisms deliver interrupts by performing Memory Write transactions. MSI and MSI-X are edge-triggered interrupt mechanisms; neither [PCI] nor this specification support level-triggered MSI/MSI-X interrupts. Certain PCI devices and their drivers rely on INTx-type level-triggered interrupt behavior (addressed by the PCI Express legacy INTx emulation mechanism). To take advantage of the MSI or MSI-X capability and edge-triggered interrupt semantics, these devices and their drivers may have to be redesigned.

MSI and MSI-X each support Per-Vector Masking (PVM). PVM is an optional¹¹³ extension to MSI, and a standard feature with MSI-X. A Function that supports the PVM extension to MSI is backward compatible with system software that is unaware of the extension. MSI-X also supports a Function Mask bit, which when Set masks all of the vectors associated with a Function.

A Legacy Endpoint that implements MSI is required to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure. A PCI Express Endpoint that implements MSI is required to support the 64-bit Message Address version of the MSI Capability structure.

The Requester of an MSI/MSI-X transaction must set the No Snoop and Relaxed Ordering attributes of the Transaction Descriptor to 0b. A Requester of an MSI/MSI-X transaction is permitted to Set the ID-Based Ordering (IDO) attribute if use of the IDO attribute is enabled.

Note that, unlike INTx emulation Messages, MSI/MSI-X transactions are not restricted to the TC0 traffic class.

IMPLEMENTATION NOTE: SYNCHRONIZATION OF DATA TRAFFIC AND MESSAGE SIGNALLED INTERRUPTS §

MSI/MSI-X transactions are permitted to use the TC that is most appropriate for the device's programming model. This is generally the same TC as is used to transfer data; for legacy I/O, TC0 should be used.

If a device uses more than one TC, it must explicitly ensure that proper synchronization is maintained between data traffic and interrupt Message(s) not using the same TC. Methods for ensuring this synchronization are implementation specific. One option is for a device to issue a zero-length Read (as described in § Section 2.2.5) using each additional TC used for data traffic prior to issuing the MSI/MSI-X transaction. Other methods are also possible. Note, however, that platform software (e.g., a device driver) is generally only capable of issuing transactions using TC0.

Within a device, different Functions are permitted to implement different sets of the MSI/MSI-X/INTx interrupt mechanisms, and system software manages each Function's interrupt mechanisms independently.

6.1.4.1 MSI Configuration §

In this section, all register and field references are in the context of the MSI Capability structure.

System software reads the Message Control register to determine the Function's MSI capabilities.

System software reads the Multiple Message Capable field (bits 3-1 of the Message Control register) to determine the number of requested vectors. MSI supports a maximum of 32 vectors per Function. System software writes to the Multiple Message Enable field (bits 6-4 of the Message Control register) to allocate either all or a subset of the requested vectors. For example, a Function can request four vectors and be allocated either four, two, or one vector. The number of

¹¹³. Exception: Within an ↓↓SR-IOV Device, ↑↑SR-IOV Device, any PFs or VFs that implement MSI must implement MSI PVM.

vectors requested and allocated is aligned to a power of two (that is, a Function that requires three vectors must request four).

If the **↓↑Per-Vector Masking Capable↓↑Per-Vector Masking Capable** bit (bit 8 of the Message Control register) is Set and system software supports Per-Vector Masking, system software may mask one or more vectors by writing to the Mask Bits register.

If the 64-bit Address Capable bit (bit 7 of the Message Control register) is Set, system software initializes the MSI Capability structure's Message Address register (specifying the lower 32 bits of the message address) and the Message Upper Address register (specifying the upper 32 bits of the message address) with a system-specified message address. System software may program the Message Upper Address register to zero so that the Function uses a 32-bit address for the MSI transaction. If this bit is Clear, system software initializes the MSI Capability structure's Message Address register (specifying a 32-bit message address) with a system specified message address.

System software initializes the MSI Capability structure's Message Data register with the lower 16 bits of a system specified data value. When the Extended Message Data Capable bit is Clear, care must be taken to initialize only the Message Data register (i.e., a 2-byte value) and not modify the upper two bytes of that DWORD location.

If the Extended Message Data Capable bit is Set and system software supports 32-bit vector values, system software may initialize the MSI capability structure's Extended Message Data register with the upper 16 bits of a system specified data value, and then Set the Extended Message Data Enable bit.

6.1.4.2 MSI-X Configuration §

In this section, all register and field references are in the context of the MSI-X Capability, MSI-X Table, and MSI-X PBA structures.

System software allocates address space for the Function's standard set of Base Address registers and sets the registers accordingly. One of the Function's Base Address registers includes address space for the MSI-X Table, though the system software that allocates address space does not need to be aware of which Base Address register this is, or the fact the address space is used for the MSI-X Table. The same or another Base Address register includes address space for the MSI-X PBA, and the same point regarding system software applies.

Depending upon system software policy, system software, device driver software, or each at different times or environments may configure a Function's MSI-X Capability and table structures with suitable vectors. For example, a booting environment will likely require only a single vector, whereas a normal operating system environment for running applications may benefit from multiple vectors if the Function supports an MSI-X Table with multiple entries. For the remainder of this section, "software" refers to either system software or device driver software.

Software reads the Table Size field from the Message Control register to determine the MSI-X Table size. The field encodes the number of table entries as N-1, so software must add 1 to the value read from the field to calculate the number of table entries N. MSI-X supports a maximum table size of 2048 entries.

Software calculates the base address of the MSI-X Table by reading the 32-bit value from the Table Offset/Table BIR register, masking off the lower 3 Table BIR bits, and adding the remaining QWORD-aligned 32-bit Table offset to the address taken from the Base Address register indicated by the Table BIR. Software calculates the base address of the MSI-X PBA using the same process with the PBA Offset/PBA BIR register.

For each MSI-X Table entry that will be used, software fills in the Message Address field, Message Upper Address field, Message Data field, and Vector Control field. The Vector Control field may contain optional Steering Tag fields. Software must not modify the Address, Data, or Steering Tag fields of an entry while it is unmasked. Refer to § [Section 6.1.4.5](#) for details.

IMPLEMENTATION NOTE: SPECIAL CONSIDERATIONS FOR QWORD ACCESSES §

Software is permitted to fill in MSI-X Table entry DWORD fields individually with DWORD writes, or software in certain cases is permitted to fill in appropriate pairs of DWORDs with a single QWORD write. Specifically, software is always permitted to fill in the Message Address and Message Upper Address fields with a single QWORD write. If a given entry is currently masked (via its Mask bit or the Function Mask bit), software is permitted to fill in the Message Data and Vector Control fields with a single QWORD write, taking advantage of the fact the Message Data field is guaranteed to become visible to hardware no later than the Vector Control field. However, if software wishes to mask a currently unmasked entry (without Setting the Function Mask bit), software must Set the entry's Mask bit using a DWORD write to the Vector Control field, since performing a QWORD write to the Message Data and Vector Control fields might result in the Message Data field being modified before the Mask bit in the Vector Control field becomes Set.

For potential use by future specifications, the Reserved bits in the Vector Control field must have their default values preserved by software. If software does not preserve their values, the result is undefined.

For each MSI-X Table entry that software chooses not to configure for generating messages, software can simply leave the entry in its default state of being masked.

Software is permitted to configure multiple MSI-X Table entries with the same vector, and this may indeed be necessary when fewer vectors are allocated than requested.

IMPLEMENTATION NOTE: HANDLING MSI-X VECTOR SHORTAGES §

For the case where fewer vectors are allocated to a Function than desired, software-controlled aliasing as enabled by MSI-X is one approach for handling the situation. For example, if a Function supports five queues, each with an associated MSI-X table entry, but only three vectors are allocated, the Function could be designed for software still to configure all five table entries, assigning one or more vectors to multiple table entries. Software could assign the three vectors {A,B,C} to the five entries as ABCCC, ABBCC, ABCBA, or other similar combinations.

Alternatively, the Function could be designed for software to configure it (using a device specific mechanism) to use only three queues and three MSI-X table entries. Software could assign the three vectors {A,B,C} to the five entries as ABC-, A-B-C, A-CB, or other similar combinations.

6.1.4.3 Enabling Operation §

To maintain backward compatibility, the MSI Enable bit in the Message Control Register for MSI and the MSI-X Enable bit in the Message Control Register for MSI-X are each Clear by default (MSI and MSI-X are both disabled). System configuration software Sets one of these bits to enable either MSI or MSI-X, but never both simultaneously. Behavior is undefined if both MSI and MSI-X are enabled simultaneously. Software disabling either mechanism during active operation may result in the Function dropping pending interrupt conditions or failing to recognize new interrupt conditions. While enabled for MSI or MSI-X operation, a Function is prohibited from using INTx interrupts (if implemented) to request service (MSI, MSI-X, and INTx are mutually exclusive).

6.1.4.4 Sending Messages §

Once MSI or MSI-X is enabled (the appropriate bit in one of the Message Control registers is Set), and one or more vectors is unmasked, the Function is permitted to send messages. To send a message, a Function does a DWORD Memory Write to the appropriate message address with the appropriate message data.

For MSI when the Extended Message Data Enable bit is Clear, the DWORD that is written is made up of the value in the MSI Message Data register in the lower two bytes and zeros in the upper two bytes. For MSI when the Extended Message Data Enable bit is Set, the DWORD that is written is made up of the value in the MSI Message Data register in the lower two bytes and the value in the MSI Extended Message Data register in the upper two bytes.

For MSI, if the Multiple Message Enable field (bits 6-4 of the Message Control Register for MSI) is non-zero, the Function is permitted to modify the low order bits of the message data to generate multiple vectors. For example, a Multiple Message Enable encoding of 010b indicates the Function is permitted to modify message data bits 1 and 0 to generate up to four unique vectors. If the Multiple Message Enable field is 000b, the Function is not permitted to modify the message data.

For MSI-X, the MSI-X Table contains at least one entry for every allocated vector, and the 32-bit Message Data field value from a selected table entry is used in the message without any modification to the low-order bits by the Function.

How a Function uses multiple vectors (when allocated) is device dependent. A Function must handle being allocated fewer vectors than requested.

6.1.4.5 Per-vector Masking and Function Masking §

Per-Vector Masking (PVM) is an optional¹¹⁴ feature with MSI, and a standard feature in MSI-X.

Function Masking is a standard feature in MSI-X. When the MSI-X Function Mask bit is Set, all of the Function's entries must behave as being masked, regardless of the per-entry Mask bit values. Function Masking is not supported in MSI, but software can readily achieve a similar effect by Setting all MSI Mask bits using a single DWORD write.

PVM in MSI-X is controlled by a Mask bit in each MSI-X Table entry. While more accurately termed “per-entry masking”, masking an MSI-X Table entry is still referred to as “vector masking” so similar descriptions can be used for both MSI and MSI-X. However, since software is permitted to program the same vector (a unique Address/Data pair) into multiple MSI-X table entries, all such entries must be masked in order to guarantee the Function will not send a message using that Address/Data pair.

For MSI and MSI-X, while a vector is masked, the Function is prohibited from sending the associated message, and the Function must Set the associated Pending bit whenever the Function would otherwise send the message. When software unmasks a vector whose associated Pending bit is Set, the Function must schedule sending the associated message, and Clear the Pending bit as soon as the message has been sent. Note that Clearing the MSI-X Function Mask bit may result in many messages needing to be sent.

If a masked vector has its Pending bit Set, and the associated underlying interrupt events are somehow satisfied (usually by software though the exact manner is Function-specific), the Function must Clear the Pending bit, to avoid sending a spurious interrupt message later when software unmasks the vector. However, if a subsequent interrupt event occurs while the vector is still masked, the Function must again Set the Pending bit.

Software is permitted to mask one or more vectors indefinitely, and service their associated interrupt events strictly based on polling their Pending bits. A Function must Set and Clear its Pending bits as necessary to support this “pure polling” mode of operation.

¹¹⁴. Exception: Within an [SR-IOV Device](#), any PFs or VFs that implement MSI must implement MSI PVM.

For MSI-X, a Function is permitted to cache Address and Data values from unmasked MSI-X Table entries. However, anytime software unmasks a currently masked MSI-X Table entry either by Clearing its Mask bit or by Clearing the Function Mask bit, the Function must update any Address or Data values that it cached from that entry. If software changes the Address or Data value of an entry while the entry is unmasked, the result is undefined.

IMPLEMENTATION NOTE: PER VECTOR MASKING WITH MSI/MSI-X §

Devices and drivers that use MSI or MSI-X have the challenge of coordinating exactly when new interrupt messages are generated. If hardware fails to send an interrupt message that software expects, an interrupt event might be “lost”. If hardware sends an interrupt message that software is not expecting, a “spurious” interrupt might result.

Per-Vector Masking (PVM) can be used to assist in this coordination. For example, when a software interrupt service routine begins, it can mask the vector to help avoid “spurious” interrupts. After the interrupt service routine services all the interrupt conditions that it is aware of, it can unmask the vector. If any interrupt conditions remain, hardware is required to generate a new interrupt message, guaranteeing that no interrupt events are lost.

PVM is a standard feature with MSI-X and an optional¹¹⁵ feature for MSI. For devices that implement MSI, implementing PVM as well is strongly recommended.

6.1.4.6 Hardware/Software Synchronization §

If a Function sends messages with the same vector multiple times before being acknowledged by software, only one message is guaranteed to be serviced. If all messages must be serviced, a device driver handshake is required. In other words, once a Function sends Vector A, it cannot send Vector A again until it is explicitly enabled to do so by its device driver (provided all messages must be serviced). If some messages can be lost, a device driver handshake is not required. For Functions that support multiple vectors, a Function can send multiple unique vectors and is guaranteed that each unique message will be serviced. For example, a Function can send Vector A followed by Vector B without any device driver handshake (both Vector A and Vector B will be serviced).

¹¹⁵. Exception: Within an [↓↓SR-IOV Device](#), [↑↑SR-IOV Device](#), any PFs or VFs that implement MSI must implement MSI PVM

IMPLEMENTATION NOTE: SERVICING MSI AND MSI-X INTERRUPTS §

When system software allocates fewer MSI or MSI-X vectors to a Function than it requests, multiple interrupt sources within the Function, each desiring a unique vector, may be required to share a single vector. Without proper handshakes between hardware and software, hardware may send fewer messages than software expects, or hardware may send what software considers to be extraneous messages.

A rather sophisticated but resource-intensive approach is to associate a dedicated event queue with each allocated vector, with producer and consumer pointers for managing each event queue. Such event queues typically reside in host memory. The Function acts as the producer and software acts as the consumer. Multiple interrupt sources within a Function may be assigned to each event queue as necessary. Each time an interrupt source needs to signal an interrupt, the Function places an entry on the appropriate event queue (assuming there's room), updates a copy of the producer pointer (typically in host memory), and sends an interrupt message with the associated vector when necessary to notify software that the event queue needs servicing. The interrupt service routine for a given event queue processes all entries it finds on its event queue, as indicated by the producer pointer. Each event queue entry identifies the interrupt source and possibly additional information about the nature of the event. The use of event queues and producer/consumer pointers can be used to guarantee that interrupt events won't get dropped when multiple interrupt sources are forced to share a vector. There's no need for additional handshaking between sending multiple messages associated with the same event queue, to guarantee that every message gets serviced. In fact, various standard techniques for "interrupt coalescing" can be used to avoid sending a separate message for every event that occurs, particularly during heavy bursts of events.

In more modest implementations, the hardware design of a Function's MSI or MSI-X logic sends a message any time a transition to assertion would have occurred on the virtual INTx wire if MSI or MSI-X had not been enabled. For example, consider a scenario in which two interrupt events (possibly from distinct interrupt sources within a Function) occur in rapid succession. The first event causes a message to be sent. Before the interrupt service routine has had an opportunity to service the first event, the second event occurs. In this case, only one message is sent, because the first event is still active at the time the second event occurs (a virtual INTx wire signal would have had only one transition to assertion).

One handshake approach for implementations like the above is to use standard Per-Vector Masking, and allow multiple interrupt sources to be associated with each vector. A given vector's interrupt service routine Sets the vector's Mask bit before it services any associated interrupting events and Clears the Mask bit after it has serviced all the events it knows about. (This could be any number of events.) Any occurrence of a new event while the Mask bit is Set results in the Pending bit being Set. If one or more associated events are still pending at the time the vector's Mask bit is Cleared, the Function immediately sends another message.

A handshake approach for MSI Functions that do not implement Per-Vector Masking is for a vector's interrupt service routine to re-inspect all of the associated interrupt events after Clearing what is presumed to be the last pending interrupt event. If another event is found to be active, it is serviced in the same interrupt service routine invocation, and the complete re-inspection is repeated until no pending events are found. This ensures that if an additional interrupting event occurs before a previous interrupt event is Cleared, whereby the Function does not send an additional interrupt message, that the new event is serviced as part of the current interrupt service routine invocation.

This alternative has the potential side effect of one vector's interrupt service routine processing an interrupting event that has already generated a new interrupt message. The interrupt service routine invocation resulting from the new message may find no pending interrupt events. Such occurrences are sometimes referred to as spurious interrupts, and software using this approach must be prepared to tolerate them.

An MSI or MSI-X message, by virtue of being a Posted Request, is prohibited by transaction ordering rules from passing Posted Requests sent earlier by the Function. The system must guarantee that an interrupt service routine invoked as a result of a given message will observe any updates performed by Posted Requests arriving prior to that message. Thus, the interrupt service routine of a device driver is not required to read from a device register in order to ensure data consistency with previous Posted Requests. However, if multiple MSI-X Table entries share the same vector, the interrupt service routine may need to read from some device specific register to determine which interrupt sources need servicing.

6.1.4.7 Message Transaction Reception and Ordering Requirements §

As with all Memory Write transactions, the device that includes the target of the interrupt message (the interrupt receiver) is required to complete all interrupt message transactions as a Completer without requiring other transactions to complete first as a Requester. In general, this means that the message receiver must complete the interrupt message transaction independent of when the CPU services the interrupt. For example, each time the interrupt receiver receives an interrupt message, it could Set a bit in an internal register indicating that this message had been received and then complete the transaction on the bus. The appropriate interrupt service routine would later be dispatched because this bit was Set. The message receiver would not be allowed to delay the completion of the interrupt message on the bus pending acknowledgement from the processor that the interrupt was being serviced. Such dependencies can lead to deadlock when multiple devices send interrupt messages simultaneously.

Although interrupt messages remain strictly ordered throughout the PCI Express Hierarchy, the order of receipt of the interrupt messages does not guarantee any order in which the interrupts will be serviced. Since the message receiver must complete all interrupt message transactions without regard to when the interrupt was actually serviced, the message receiver will generally not maintain any information about the order in which the interrupts were received. This is true both of interrupt messages received from different devices and multiple messages received from the same device. If a device requires one interrupt message to be serviced before another, the device must not send the second interrupt message until the first one has been serviced.

6.1.5 PME Support §

PCI Express supports power management events from native PCI Express devices as well as PME-capable PCI devices.

PME signaling is accomplished using an in-band Transaction Layer PME Message (PM_PME) as described in § Chapter 5.

6.1.6 Native PME Software Model §

PCI Express-aware software can enable a mode where the Root Complex signals PME via an interrupt. When configured for native PME support, a Root Port receives the PME Message and sets the PME Status bit in its Root Status Register. If software has set the PME Interrupt Enable bit in the Root Control Register to 1b, the Root Port then generates an interrupt.

If the Root Port is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as all of the following conditions are satisfied:

- The Interrupt Disable bit in the Command register is set to 0b.
- The PME Interrupt Enable bit in the Root Control Register is set to 1b.
- The PME Status bit in the Root Status register is set.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If the Root Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- The PME Interrupt Enable bit in the Root Control Register is set to 1b.
- The PME Status bit in the Root Status Register is set.

Note that PME and Hot-Plug Event interrupts (when both are implemented) always share the same MSI or MSI-X vector, as indicated by the Interrupt Message Number field in the PCI Express Capabilities Register.

The software handler for this interrupt can determine which device sent the PME Message by reading the PME Requester ID field in the Root Status Register in a Root Port. It dismisses the interrupt by writing a 1b to the PME Status bit in the Root Status Register. Refer to § [Section 7.5.3.14](#) for more details.

Root Complex Event Collectors provide support for the above described functionality for Root Complex Integrated Endpoints (RCiEPs).

6.1.7 Legacy PME Software Model §

Legacy software, however, will not understand this mechanism for signaling PME. In the presence of legacy system software, the system power management logic in the Root Complex receives the PME Message and informs system software through an implementation specific mechanism. The Root Complex may utilize the Requester ID in the PM_PME to inform system software which device caused the power management event.

Since it is delivered by a Message, PME has edge-triggered semantics in PCI Express, which differs from the level-triggered PME mechanism used for conventional PCI. It is the responsibility of the Root Complex to abstract this difference from system software to maintain compatibility with conventional PCI systems.

6.1.8 Operating System Power Management Notification §

In order to maintain compatibility with non-PCI Express-aware system software, system power management logic must be configured by firmware to use the legacy mechanism of signaling PME by default. PCI Express-aware system software must notify the firmware prior to enabling native, interrupt-based PME signaling. In response to this notification, system firmware must, if needed, reconfigure the Root Complex to disable legacy mechanisms of signaling PME. The details of this firmware notification are beyond the scope of this specification, but since it will be executed at system run-time, the response to this notification must not interfere with system software. Therefore, following control handoff to the operating system, firmware must not write to available system memory or any PCI Express resources (e.g., Configuration Space structures) owned by the operating system.

6.1.9 PME Routing Between PCI Express and PCI Hierarchies §

PME-capable conventional PCI and PCI-X devices assert the PME# pin to signal a power management event. The PME# signal from PCI or PCI-X devices may either be converted to a PCI Express in-band PME Message by a PCI Express-PCI Bridge or routed directly to the Root Complex.

If the PME# signal from a PCI or PCI-X device is routed directly to the Root Complex, it signals system software using the same mechanism used in present PCI systems. A Root Complex may optionally provide support for signaling PME from PCI or PCI-X devices to system software via an interrupt. In this scenario, it is recommended for the Root Complex to detect the Bus, Device and Function Number of the PCI or PCI-X device that asserted PME#, and use this information to

fill in the PME Requester ID field in the Root Port that originated the hierarchy containing the PCI or PCI-X device. If this is not possible, the Root Complex may optionally write the Requester ID of the Root Port to this field.

Since RCiEPs are not contained in any of the hierarchy domains originated by Root Ports, RCiEPs not associated with a Root Complex Event Collector signal system software of a PME using the same mechanism used in present PCI systems. A Root Complex Event Collector, if implemented, enables the PCI Express Native PME model for associated RCiEPs.

6.2 Error Signaling and Logging §

In this document, errors which must be checked and errors which may optionally be checked are identified. Each such error is associated either with the Port or with a specific device (or Function in a Multi-Function Device), and this association is given along with the description of the error. This section will discuss how errors are classified and reported.

6.2.1 Scope §

This section explains the error signaling and logging requirements for PCI Express components. This includes errors which occur on the PCI Express interface itself, those errors which occur on behalf of transactions initiated on PCI Express, and errors which occur within a component and are related to the PCI Express interface. This section does not focus on errors which occur within the component that are unrelated to a PCI Express interface. This type of error signaling is better handled through proprietary methods employing device-specific interrupts.

PCI Express defines two error reporting paradigms: baseline capability and [Advanced Error Reporting \(AER\) capability](#). [Advanced Error Reporting Extended Capability \(AER\)](#) Baseline capability is required for all PCI Express devices, and it defines the minimum error reporting requirements. [AER capability](#) [AER Extended Capability](#) defines more robust error reporting and is implemented with a specific PCI Express Capability structure (refer to § Section 7.8.4 for a definition of this optional capability). This section explicitly calls out all error handling differences between baseline and [AER capability](#).

All [SR-IOV devices](#) must support baseline Capability, with certain modifications to account for the goal of reduced cost and complexity of implementation. [AER capability](#) is optional, but if [AER](#) is not implemented in a PF, it must not be implemented in its associated VFs. If [AER](#) is implemented in the PF, it is optional in its VFs.

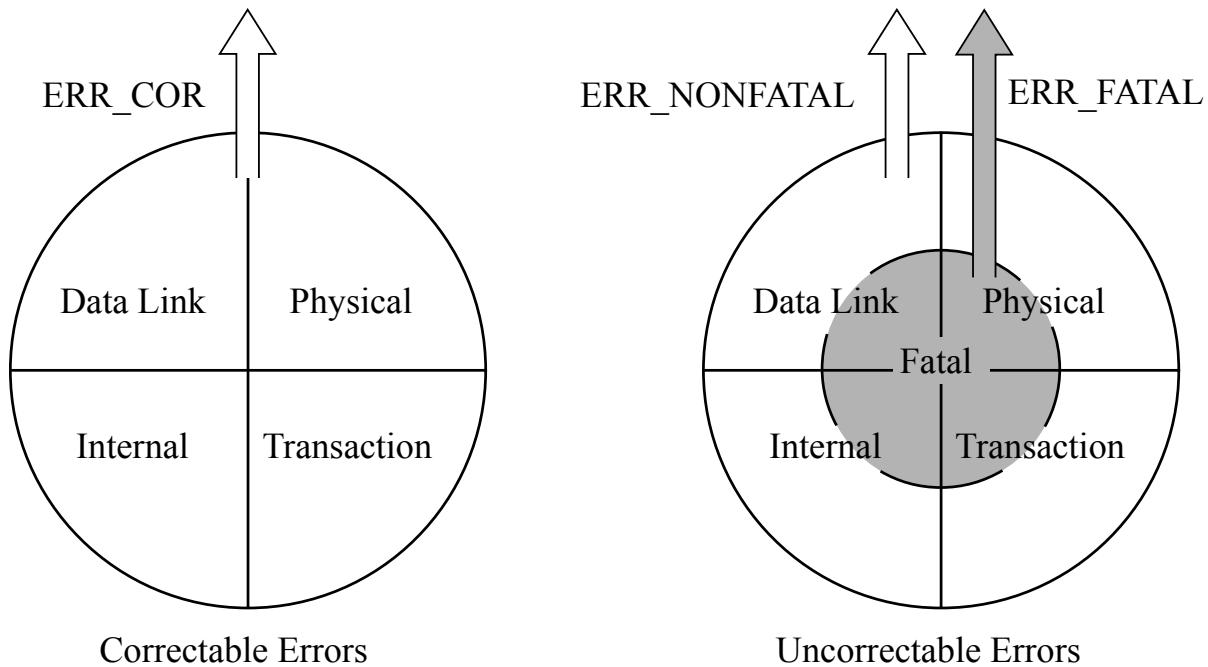
[All SIOV Devices must support baseline Capability, with certain modifications to account for the goal of reduced cost and complexity of implementation. The AER Extended Capability is optional.](#)

ECN: Base 6.3
SIOV△

All PCI Express devices support existing, non-PCI Express-aware, software for error handling by mapping PCI Express errors to existing PCI reporting mechanisms, in addition to the PCI Express-specific mechanisms.

6.2.2 Error Classification §

PCI Express errors can be classified as two types: Uncorrectable errors and Correctable errors. This classification separates those errors resulting in functional failure from those errors resulting in degraded performance. Uncorrectable errors can further be classified as Fatal or Non-Fatal (see § Figure 6-1).



OM13827A

Figure 6-1 Error Classification

Classification of error severity as Fatal, Uncorrectable, and Correctable provides the platform with mechanisms for mapping the error to a suitable handling mechanism. For example, the platform might choose to respond to correctable errors with low priority, performance monitoring software. Such software could count the frequency of correctable errors and provide Link integrity information. On the other hand, a platform designer might choose to map Fatal errors to a system-wide reset. It is the decision of the platform designer to map these PCI Express severity levels onto platform level severities.

6.2.2.1 Correctable Errors

Correctable errors include those error conditions where hardware can recover without any loss of information. Hardware corrects these errors and software intervention is not required. For example, an LCRC error in a TLP that might be corrected by Data Link Level Retry is considered a correctable error. Measuring the frequency of Link-level correctable errors may be helpful for profiling the integrity of a Link.

Correctable errors also include transaction-level cases where one agent detects an error with a TLP, but another agent is responsible for taking any recovery action if needed, such as re-attempting the operation with a separate subsequent transaction. The detecting agent can be configured to report the error as being correctable since the recovery agent may be able to correct it. If recovery action is indeed needed, the recovery agent must report the error as uncorrectable if the recovery agent decides not to attempt recovery.

The triggering of Downstream Port Containment (DPC) is not handled as an error, but it can be signaled as if it were a correctable error, since software that takes advantage of DPC can sometimes recover from the uncorrectable error that triggered DPC. See § Section 6.2.11 . An ERR_COR Message that's used for DPC signaling is intended to target system firmware, and may indicate so via the ERR_COR Subclass field.

Similarly, ERR_COR may be used by the System Firmware Intermediary (SFI) capability to signal system firmware, and must indicate so via the ERR_COR Subclass field. See § Section 6.7.4 .

IMPLEMENTATION NOTE: ERR_COR SUBCLASS POSSIBLE USAGE §

↑↑The following is a high-level summary of how the ERR_COR Subclass (ECS) field can be used↑

ECN: Base 6.3
RC-ECS△<▷

- ↑↑DPC events: DPC Trigger Status, DL_Active↑
 - ↑↑ECS SIG_SFW – ERR_CORs intended to signal SFW:↑
 - ↑↑SFI events: DL_Active, SFI_DRS↑
- ↑↑ECS SIG_OS – ERR_CORs intended to signal the OS:↑
 - ↑↑AER events: Link correctable errors, Advisory Non-Fatal Errors↑
 - ↑↑RP PIO events: RP PIO advisory errors↑
- ↑↑ECS Legacy – ERR_CORs from adapters that do not support ECS↑
 - ↑↑Implementation-specific mechanisms may enable Legacy ERR_CORs to signal SFW↑

6.2.2.2 Uncorrectable Errors §

Uncorrectable errors are those error conditions that impact functionality of the interface. There is no mechanism defined in this specification to correct these errors. Reporting an uncorrectable error is analogous to asserting SERR# in PCI/PCI-X. For more robust error handling by the system, this specification further classifies uncorrectable errors as Fatal and Non-fatal.

6.2.2.2.1 Fatal Errors §

Fatal errors are uncorrectable error conditions which render the particular Link and related hardware unreliable. For Fatal errors, a reset of the components on the Link may be required to return to reliable operation. Platform handling of Fatal errors, and any efforts to limit the effects of these errors, is platform implementation specific.

6.2.2.2.2 Non-Fatal Errors §

Non-fatal errors are uncorrectable errors which cause a particular transaction to be unreliable but the Link is otherwise fully functional. Isolating Non-fatal from Fatal errors provides Requester/Receiver logic in a device or system management software the opportunity to recover from the error without resetting the components on the Link and disturbing other transactions in progress. Devices not associated with the transaction in error are not impacted by the error.

6.2.3 Error Signaling §

There are three complementary mechanisms which allow the agent detecting an error to alert the system or another device that an error has occurred. The first mechanism is through a Completion Status, the second method is with in-band error Messages, and the third is with Error Forwarding (also known as data poisoning).

Note that it is the responsibility of the agent detecting the error to signal the error appropriately.

§ Section 6.2.7 describes all the errors and how the hardware is required to respond when the error is detected.

6.2.3.1 Completion Status §

The Completion Status field (when status is not Successful Completion) in the Completion header indicates that the associated Request failed (see § Section 2.2.8.10). This is one method of error reporting which enables the Requester to associate an error with a specific Request. In other words, since Non-Posted Requests ^{↑↑and UIO}
Requests[↑] are not considered complete until after the Completion returns, the Completion Status field gives the Requester an opportunity to “fix” the problem at some higher level protocol (outside the scope of this specification).

Errata: Base 6.3
B834Δ<|>

6.2.3.2 Error Messages §

Error Messages are sent to the Root Complex for reporting the detection of errors according to the severity of the error.

Error messages that originate from PCI Express or Legacy Endpoints are sent to corresponding Root Ports. Errors that originate from a Root Port itself are reported through the same Root Port.

If an optional Root Complex Event Collector is implemented, errors that originate from RCiEPs are sent to the corresponding Root Complex Event Collector. Errors that originate in a Root Complex Event Collector itself are reported through the same Root Complex Event Collector. The Root Complex Event Collector must declare supported RCiEPs as part of its capabilities; each RCiEP must be associated with no more than one Root Complex Event Collector.

When multiple errors of the same severity are detected, the corresponding error Messages with the same Requester ID may be merged for different errors of the same severity. At least one error Message must be sent for detected errors of each severity level. Note, however, that the detection of a given error in some cases will preclude the reporting of certain errors. Refer to § Section 6.2.3.2.3. Also note special rules in § Section 6.2.4 regarding non-Function-specific errors in Multi-Function Devices.

Table 6-1 Error Messages §

| Error Message | Description |
|---------------|--|
| ERR_COR | This Message is issued when the Function or Device detects a correctable error on the PCI Express interface. Refer to § Section 6.2.2.1 for the definition of a correctable error. |
| ERR_NONFATAL | This Message is issued when the Function or Device detects a Non-fatal, uncorrectable error on the PCI Express interface. Refer to § Section 6.2.2.2 for the definition of a Non-fatal, uncorrectable error. |
| ERR_FATAL | This Message is issued when the Function or Device detects a Fatal, uncorrectable error on the PCI Express interface. Refer to § Section 6.2.2.2.1 for the definition of a Fatal, uncorrectable error. |

For these Messages, the Root Complex identifies the initiator of the Message by the Requester ID of the Message header. The Root Complex translates these error Messages into platform level events.

IMPLEMENTATION NOTE: USE OF ERR_COR, ERR_NONFATAL, AND ERR_FATAL §

In [PCIe-1.0a], a given error was either correctable, non-fatal, or fatal. Assuming signaling was enabled, correctable errors were always signaled with ERR_COR, non-fatal errors were always signaled with ERR_NONFATAL, and fatal errors were always signaled with ERR_FATAL.

In subsequent specifications that support Role-Based Error Reporting, non-fatal errors are sometimes signaled with ERR_NONFATAL, sometimes signaled with ERR_COR, and sometimes not signaled at all, depending upon the role of the agent that detects the error and whether the agent implements AER (see § Section 6.2.3.2.4). On some platforms, sending ERR_NONFATAL will preclude another agent from attempting recovery or determining the ultimate disposition of the error. For cases where the detecting agent is not the appropriate agent to determine the ultimate disposition of the error, a detecting agent with AER can signal the non-fatal error with ERR_COR, which serves as an advisory notification to software. For cases where the detecting agent is the appropriate one, the agent signals the non-fatal error with ERR_NONFATAL.

For a given uncorrectable error that's normally non-fatal, if software wishes to avoid continued hierarchy operation upon the detection of that error, software can configure detecting agents that implement AER to escalate the severity of that error to fatal. A detecting agent (if enabled) will always signal a fatal error with ERR_FATAL, regardless of the agent's role.

Software should recognize that a single transaction can be signaled by multiple agents using different types of error Messages. For example, a poisoned TLP might be signaled by intermediate Receivers with ERR_COR, while the ultimate destination Receiver might signal it with ERR_NONFATAL.

6.2.3.2.1 Uncorrectable Error Severity Programming (Advanced Error Reporting) §

For device Functions implementing the Advanced Error Reporting Extended Capability, the Uncorrectable Error Severity register allows each uncorrectable error to be programmed to Fatal or Non-Fatal. Uncorrectable errors are not recoverable using defined PCI Express mechanisms. However, some platforms or devices might consider a particular error fatal to a Link or device while another platform considers that error non-fatal. The default value of the Uncorrectable Error Severity register serves as a starting point for this specification but the register can be reprogrammed if the device driver or platform software requires more robust error handling.

Baseline error handling does not support severity programming.

6.2.3.2.2 Masking Individual Errors §

§ Section 6.2.7 lists all the errors governed by this specification and describes when each of the above error Messages are issued. The transmission of these error Messages by class (correctable, non-fatal, fatal) is enabled using the Reporting Enable bits of the Device Control register (see § Section 7.5.3.4) or the SERR# Enable bit in the PCI Command register (see § Section 7.5.1.1.3).

For devices implementing the Advanced Error Reporting Extended Capability the Uncorrectable Error Mask register and Correctable Error Mask register allows each error condition to be masked independently. If Messages for a particular class of error are not enabled by the combined settings in the Device Control register and the PCI Command register, then no Messages of that class will be sent regardless of the values for the corresponding mask register.

If an individual error is masked when it is detected, its error status bit is still affected, but no error reporting Message is sent to the Root Complex, and the error is not recorded in the Header Log, TLP Prefix Log, or First Error Pointer.

6.2.3.2.3 Error Pollution §

Error pollution can occur if error conditions for a given transaction are not isolated to the most significant occurrence. For example, assume the Physical Layer detects a Receiver Error . This error is detected at the Physical Layer and an error is reported to the Root Complex. To avoid having this error propagate and cause subsequent errors at upper layers (for example, a TLP error at the Data Link Layer), making it more difficult to determine the root cause of the error, subsequent errors which occur for the same packet will not be reported by the Data Link or Transaction layers. Similarly, when the Data Link Layer detects an error, subsequent errors which occur for the same packet will not be reported by the Transaction Layer. This behavior applies only to errors that are associated with a particular packet - other errors are reported for each occurrence.

Corrected Internal Errors are errors whose effect has been masked or worked around by a component; refer to § Section 6.2.10 for details. Therefore, Corrected Internal Errors do not contribute to error pollution and should be reported when detected.

For errors detected in the Transaction layer and Uncorrectable Internal Errors , it is permitted and recommended that no more than one error be reported for a single received TLP, and that the following precedence (from highest to lowest) be used:

- Uncorrectable Internal Error
- Receiver Overflow
- Malformed TLP
- IDE Check Failed
- ECRC Check Failed
- Misrouted IDE TLP
- AtomicOp, DMWr, or TLP Translation Egress Blocked
- TLP Prefix Blocked
- ACS Violation
- MC Blocked TLP
- Unsupported Request (UR), Completer Abort (CA), or Unexpected Completion
- PCRC Check Failed
- Poisoned TLP Received or Poisoned TLP Egress Blocked

The Completion Timeout error is not in the above precedence list, since it is not detected by processing a received TLP. Errors listed under the same bullet are mutually exclusive, so their relative order does not matter.

6.2.3.2.4 Advisory Non-Fatal Error Cases §

In some cases the detector of a non-fatal error is not the most appropriate agent to determine whether the error is recoverable or not, or if it even needs any recovery action at all. For example, if software attempts to perform a configuration read from a non-existent device or Function, the resulting UR Status in the Completion will signal the error to software, and software does not need for the Completer in addition to signal the error by sending an ERR_NONFATAL Message. In fact, on some platforms, signaling the error with ERR_NONFATAL results in a System Error, which breaks normal software probing.

“Advisory Non-Fatal Error” cases are predominantly determined by the role of the detecting agent (Requester, Completer, or Receiver) and the specific error. In such cases, an agent with AER signals the non-fatal error (if enabled) by

sending an ERR_COR Message as an advisory to software, instead of sending ERR_NONFATAL. An agent without AER sends no error Message for these cases, since software receiving ERR_COR would be unable to distinguish Advisory Non-Fatal Error cases from the correctable error cases used to assess Link integrity.

Following are the specific cases of Advisory Non-Fatal Errors. Note that multiple errors from the same or different error classes (correctable, non-fatal, fatal) may be present with a single TLP. For example, an unexpected Completion might also be poisoned. Refer to § Section 6.2.3.2.3 for requirements and recommendations on reporting multiple errors. For the previous example, it is recommended that Unexpected Completion be reported, and that Poisoned TLP Received not be reported.

If software wishes for an agent with AER to handle what would normally be an Advisory Non-Fatal Error case as being more serious, software can escalate the severity of the uncorrectable error to fatal, in which case the agent (if enabled) will signal the error with ERR_FATAL.

This section covers Advisory Non-Fatal Error handling for errors managed by the PCI Express Extended Capability and AER. § Section 6.2.11.3 covers the RP PIO error handling mechanism for Root Ports that support RP Extensions for DPC. RP PIO advisory errors are similar in concept to AER Advisory Non-Fatal Errors, but apply to different error cases and are managed by different controls.

6.2.3.2.4.1 Completer Sending a Completion with UR/CA Status §

A Completer generally sends a Completion with an Unsupported Request or Completer Abort (UR/CA) Status to signal an uncorrectable error for a Non-Posted ↑↑Request or UIO↑ Request.¹¹⁶ If the severity of the UR/CA error¹¹⁷ is non-fatal, the Completer must handle this case as an Advisory Non-Fatal Error.¹¹⁸ A Completer with AER signals the non-fatal error (if enabled) by sending an ERR_COR Message. A Completer without AER sends no error Message for this case.

 Errata: Base 6.3
B834Δ◀▶

Even though there was an uncorrectable error for this specific transaction, the Completer must handle this case as an Advisory Non-Fatal Error, since the Requester upon receiving the Completion with UR/CA Status is responsible for reporting the error (if necessary) using a Requester-specific mechanism (see § Section 6.2.3.2.5).

6.2.3.2.4.2 Intermediate Receiver §

When a Receiver that's not serving as the ultimate PCI Express destination for a TLP detects¹¹⁹ a non-fatal error with the TLP, this "intermediate" Receiver must handle this case as an Advisory Non-Fatal Error.¹²⁰ A Receiver with AER signals the error (if enabled) by sending an ERR_COR Message. A Receiver without AER sends no error Message for this case. An exception to the intermediate Receiver case for Root Complexes (RCs) is noted below.

An example where the intermediate Receiver case occurs is a Switch that detects poison or bad ECRC in a TLP that it is routing. Even though this was an uncorrectable (but non-fatal) error at this point in the TLP's route, the intermediate Receiver handles it as an Advisory Non-Fatal Error, so that the ultimate Receiver of the TLP (i.e., the Completer for a Request TLP, or the Requester for a Completion TLP) is not precluded from handling the error more appropriately according to its error settings. For example, a given Completer that detects poison in a Memory Write Request¹²¹ might have the error masked (and thus go unsignaled), whereas a different Completer in the same hierarchy might signal that error with ERR_NONFATAL.

116. If the Completer is returning data in a Completion, and the data is bad or suspect, the Completer is permitted to signal the error using the Error Forwarding (Data Poisoning) mechanism instead of handling it as a UR or CA.

117. Certain other errors (e.g., ACS Violation) with a Non-Posted Request ↑↑Request or UIO↑ also result in the Completer sending a Completion with UR or CA Status. If the severity of the error (e.g., ACS Violation) is non-fatal, the Completer must also handle this case as an Advisory Non-Fatal Error. However, see § Section 2.7.2.1 regarding certain Requests with Poisoned data that must be handled as uncorrectable errors.

118. If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

119. If the Receiver does not implement ECRC Checking or ECRC Checking is not enabled, the Receiver will not detect an ECRC Error.

120. If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

121. See § Section 2.7.2.1 for special rules that apply for poisoned Memory Write Requests.

A Poisoned TLP Egress Blocked error is never handled as an intermediate Receiver case since it is not detected as a part of processing a received TLP.

If an RC detects a non-fatal error with a TLP it normally would forward peer-to-peer between Root Ports, but the RC does not support propagating the error related information (e.g., a TLP Digest, EP bit, or equivalent) with the forwarded transaction, the RC must signal the error (if enabled) with ERR_NONFATAL and also must not forward the transaction. An example is an RC needing to forward a poisoned TLP peer-to-peer between Root Ports, but the RC's internal fabric does not support poison indication.

6.2.3.2.4.3 Ultimate PCI Express Receiver of a Poisoned TLP or IDE TLP with PCRC Check Failed §

When a poisoned TLP is received by its ultimate PCI Express destination, if the severity is non-fatal and the Receiver legitimately chooses not to handle this case as an uncorrectable error (see below), the Receiver must handle this case as an Advisory Non-Fatal Error.¹²² When a IDE TLP is determined at its ultimate destination Port to have a PCRC Check Failed error, if the severity is non-fatal and the Receiver deals with the ↑↓poisoned↓ data ↑↓payload↑ in a manner that permits continued operation, the Receiver must handle this case as an Advisory Non-Fatal Error. A Receiver with AER signals the error (if enabled) by sending an ERR_COR Message. A Receiver without AER sends no error Message for this case.

Errata: Base 6.3
B841△◀

A Receiver must not handle this case as an Advisory Non-Fatal Error if either of the following apply:

- Handling the error as a Correctable Error and continuing operation when configured correctly could lead to silent data corruption.
- Rules in § Section 2.7.2.1 require this case to be handled as an uncorrectable error.

An example is a Root Complex that receives a poisoned Memory Write TLP that targets host memory. If the Root Complex propagates the poisoned data along with its indication to host memory, it signals the error (if enabled) with an ERR_COR. If the Root Complex does not propagate the poison to host memory, it signals the error (if enabled) with ERR_NONFATAL.

Another example is a Requester that receives a poisoned Memory Read Completion TLP. If the Requester propagates the poisoned data internally or handles the error like it would for a Completion with UR/CA Status, it signals the error (if enabled) with an ERR_COR. If the Requester does not handle the poison in a manner that permits continued operation, it signals the error (if enabled) with ERR_NONFATAL.

6.2.3.2.4.4 Requester with Completion Timeout §

This section applies to Requesters other than Root Ports performing programmed I/O (PIO). See § Section 6.2.11.3 for related RP PIO functionality in Root Ports that support RP Extensions for DPC.

When the Requester of a Non-Posted Request ↑↓or UIO Request↑ times out while waiting for the associated Completion, the Requester is permitted to attempt to recover from the error by issuing a separate subsequent Request. The Requester is permitted to attempt recovery zero, one, or multiple (finite) times, but must signal the error (if enabled) with an uncorrectable error Message if no further recovery attempt will be made.

Errata: Base 6.3
B834△◀

If the severity of the Completion Timeout is non-fatal, and the Requester elects to attempt recovery by issuing a new request, the Requester must first handle the current error case as an Advisory Non-Fatal Error.¹²³ A Requester with AER

122. If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

123. If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL. The Requester is strongly discouraged from attempting recovery since sending ERR_FATAL will often result in the entire hierarchy going down.

signals the error (if enabled) by sending an ERR_COR Message. A Requester without AER sends no error Message for this case.

Note that automatic recovery by the Requester from a Completion Timeout is generally possible only if the Non-Posted Request ↓↓ or UIO Request↑ has no side-effects, but may also depend upon other considerations outside the scope of this specification.

Errata: Base 6.3
B834△<D

6.2.3.2.4.5 Receiver of an Unexpected Completion §

When a Receiver receives an unexpected Completion and the severity of the Unexpected Completion error is non-fatal, the Receiver must handle this case as an Advisory Non-Fatal Error.¹²⁴ A Receiver with AER signals the error (if enabled) by sending an ERR_COR Message. A Receiver without AER sends no error Message for this case.

If the unexpected Completion was a result of misrouting, the Completion Timeout mechanism at the associated Requester will trigger eventually, and the Requester may elect to attempt recovery. Interference with Requester recovery can be avoided by having the Receiver of the unexpected Completion handle the error as an Advisory Non-Fatal Error.

6.2.3.2.5 Requester Receiving a Completion with UR/CA Status §

When a Requester receives back a Completion with a UR/CA Status, generally the Completer has handled the error as an Advisory Non-Fatal Error, assuming the error severity was non-fatal at the Completer (see § Section 6.2.3.2.4.1). The Requester must determine if any error recovery action is necessary, what type of recovery action to take, and whether or not to report the error.

If the Requester needs to report the error, the Requester must do so solely through a Requester-specific mechanism. For example, many devices have an associated device driver that can report errors to software. As another important example, the Root Complex on some platforms returns all 1's to software if a Configuration Read Completion has a UR/CA Status.

§ Section 6.2.11.3 covers RP PIO controls for Root Ports that support RP Extensions for DPC. Outside of the RP PIO mechanisms, Requesters are not permitted to report the error using PCI Express logging and error Message signaling.

6.2.3.3 Error Forwarding (Data Poisoning) §

Error Forwarding, also known as data poisoning, is indicated by setting the EP bit in a TLP. Refer to § Section 2.7.2. This is another method of error reporting in PCI Express that enables the Receiver of a TLP to associate an error with a specific Request or Completion. Unlike the Completion Status mechanism, Error Forwarding can be used with either Requests or Completions that contain data. In addition, “intermediate” Receivers along the TLP’s route, not just the Receiver at the ultimate destination, are required to detect and report (if enabled) receiving the poisoned TLP. This can help software determine if a particular Switch along the path poisoned the TLP.

6.2.3.4 Optional Error Checking §

This specification contains a number of optional error checks. Unless otherwise specified, behavior is undefined if an optional error check is not performed and the error occurs.

When an optional error check involves multiple rules, unless otherwise specified, each rule is independently optional. An implementation may check against all of the rules, none of them or any combination.

¹²⁴. If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

Unless otherwise specified, implementation specific criteria are used in determining whether an optional error check is performed.

6.2.4 Error Logging §

§ Section 6.2.7 lists all the errors governed by this specification and for each error, the logging requirements are specified. Device Functions that do not support the Advanced Error Reporting Extended Capability log only the Device Status register bits indicating that an error has been detected. Note that some errors are also reported using the reporting mechanisms in the PCI-compatible (Type 00h and 01h) configuration registers. § Section 7.5.1 describes how these register bits are affected by the different types of error conditions described in this section.

For device Functions supporting the Advanced Error Reporting Extended Capability, each of the errors in § Table 6-3, § Table 6-4, and § Table 6-5 corresponds to a particular bit in the Uncorrectable Error Status register or Correctable Error Status register. These registers are used by software to determine more precisely which error and what severity occurred. For specific Transaction Layer errors and Uncorrectable Internal Errors, the associated TLP header is recorded.

In a Multi-Function Device other than an ~~↓↑SR-IOV device, ↓↑SR-IOV Device or an SIOV Device, ↑~~ PCI Express errors not specific to any single Function within the device must be logged in the corresponding status and logging registers of all Functions in that device.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

In an ~~↓↑SR-IOV device, ↓↑SR-IOV Device or an SIOV Device, ↑~~ errors identified as non-Function-specific must be logged in all ~~↓↑PFs, and not in their associated VFs, ↓↑PFs, ↑~~ Such errors must also be logged in any non-IOV Functions. ~~↓↑Non-Function-specific errors must not be logged in VFs, ↑~~

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

The following PCI Express errors are not Function-specific:

- All Physical Layer errors
- All Data Link Layer errors
- These Transaction Layer errors:
 - ECRC Check Failed
 - Unsupported Request, when caused by no Function claiming a TLP
 - Receiver Overflow
 - Flow Control Protocol Error
 - Malformed TLP
 - Unexpected Completion, when caused by no Function claiming a Completion
 - Unexpected Completion, when caused by a Completion that cannot be forwarded by a Switch, and the Ingress Port is a Switch Upstream Port associated with a Multi-Function Device
 - Some Transaction Layer errors (e.g., Poisoned TLP Received) may be Function-specific or not, depending upon whether the associated TLP targets a single Function or all Functions in that device.
- Some Internal Errors
 - The determination of whether an Internal Error is Function-specific or not is implementation specific.

On the detection of one of these errors, a Multi-Function Device should generate at most one error reporting Message of a given severity, where the Message must report the Requester ID of a Function of the device that is enabled to report that specific type of error. If no Function is enabled to send a reporting Message, the device does not send a reporting Message. If all reporting-enabled Functions have the same severity level set for the error, only one error Message is sent. If all reporting-enabled Functions do not have the same severity level set for the error, one error Message for each

severity level is sent. Software is responsible for scanning all Functions in a Multi-Function Device when it detects one of those errors.

↑↑If an SIOV Device generates the Message for an error that is attributable to an SDI , the error Message must contain the RID of the SDI . Errors attributable to a specific SDI may optionally be handled through per-SDI mechanisms that are outside the scope of this specification.↑

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

6.2.4.1 Root Complex Considerations (Advanced Error Reporting) §

6.2.4.1.1 Error Source Identification §

In addition to the above logging, a Root Port or Root Complex Event Collector that supports the Advanced Error Reporting Extended 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 ↑↑applicable¹²⁵ ERR_COR (correctable errors) Messages received by the Root Port or Root Complex Event Collector. System software written to support Advanced Error Reporting can use the Root Error Status register to determine which fields hold valid information.

If an RCIEP is associated with a Root Complex Event Collector , the RCIEP must report its errors through that Root Complex Event Collector .

For both Root Ports and Root Complex Event Collectors, in order for a received error Message or an internally generated error Message to be recorded in the Root Error Status register and the Error Source Identification register , the error Message must be “transmitted”. Refer to § Section 6.2.8.1 for information on how received Messages are forwarded and transmitted. Internally generated error Messages are enabled for transmission with the SERR# Enable bit in the Command register (ERR_NONFATAL and ERR_FATAL) or the Reporting Enable bits in the Device Control register (ERR_COR , ERR_NONFATAL , and ERR_FATAL).

6.2.4.1.2 Interrupt Generation §

The Root Error Command register allows further control of Root Complex response to Correctable, Non-Fatal, and Fatal error Messages than the basic Root Complex capability to generate system errors in response to error Messages. Bit fields enable or disable generation of interrupts for the three types of error Messages. System error generation in response to error Messages may be disabled via the PCI Express Capability structure.

If a Root Port or Root Complex Event Collector is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as all of the following conditions are satisfied:

- The Interrupt Disable bit in the Command register is set to 0b.
- At least one Error Reporting Enable bit in the Root Error Command register and its associated error Messages Received bit in the Root Error Status register are both set to 1b.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If a Root Port or Root Complex Event Collector is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).

125. ↑↑To see which ERR_COR Messages are not applicable for this section, see § Section 6.2.4.1.3.↑

- At least one Error Reporting Enable bit in the Root Error Command register and its associated error Messages Received bit in the Root Error Status register are both set to 1b.

Note that Advanced Error Reporting MSI/MSI-X interrupts always use the vector indicated by the Advanced Error Interrupt Message Number field in the Root Error Status register.

6.2.4.1.3 §

ECN: Base 6.3
RC-ECS $\triangleleft\triangleright$

 If an RP or RCEC supports RC ECS Handling as indicated by the RC ECS Handling Capable bit being Set, the Function can control individually whether each of the three ECS subclasses signals SFW versus the OS.

 There are three enable bits for directing ERR_CORs to be handled by SFW, one for each of the ECS subclasses (ECS SIG_SFW, ECS SIG_OS, and ECS Legacy). In each case, if the enable bit is Set, that subclass of ERR_COR Messages signals SFW via a system-specific mechanism; otherwise, that subclass of ERR_COR Messages signals the OS using architected interrupts. The specific bits are ECS SIG_SFW Handling by SFW Enable, ECS SIG_OS Handling by SFW Enable, and ECS Legacy Handling by SFW Enable.

 The default is for all three enable bits to be Clear, resulting in behavior identical to RPs or RCECs that do not support RC ECS Handling.

 To enable the handling of ERR_COR Messages by SFW and the OS concurrently and independently, RC ECS Handling defines different resources to be used by ERR_COR Messages being handled by SFW. These include the SFW Handling ERR_COR Received bit, the Multiple SFW Handling ERR_COR Received bit, and the SFW Handling Source ID field. These resources correspond to similar resources used for handling interrupts to the OS. See the specific resources for details on their operation.

6.2.4.2 Multiple Error Handling (Advanced Error Reporting Extended Capability) §

For the Advanced Error Reporting Extended Capability, the Uncorrectable Error Status register and Correctable Error Status register accumulate the collection of errors which correspond to that particular PCI Express interface. The bits remain set until explicitly cleared by software or reset. Since multiple bits might be set in the Uncorrectable Error Status register, the First Error Pointer (when valid) points to the oldest uncorrectable error that is recorded. The First Error Pointer is valid when the corresponding bit of the Uncorrectable Error Status register is set. The First Error Pointer is invalid when the corresponding bit of the Uncorrectable Error Status register is not set, or is an undefined bit.

The Advanced Error Reporting Extended Capability provides the ability to record headers¹²⁶ for errors that require header logging. An implementation may support the recording of multiple headers, but at a minimum must support the ability of recording at least one. The ability to record multiple headers is indicated by the state of the Multiple Header Recording Capable bit and enabled by the Multiple Header Recording Enable bit of the Advanced Error Capabilities and Control register. When multiple header recording is supported and enabled, errors are recorded in the order in which they are detected.

If no header recording resources are available when an unmasked uncorrectable error is detected, its error status bit is Set, but the error is not recorded. If an uncorrectable error is masked when it is detected, its error status bit is Set, but the error is not recorded.

When software is ready to dismiss a recorded error indicated by the First Error Pointer, software writes a 1b to the indicated error status bit to clear it, which causes hardware to free up the associated recording resources. If another instance of that error is still recorded, hardware is permitted but not required to leave that error status bit set. If any error instance is still recorded, hardware must immediately update the Header Log, TLP Prefix Log, TLP Prefix Log Present bit, First Error Pointer, and Uncorrectable Error Status register to reflect the next recorded error. If no other error

126. If a Function supports TLP Prefixes, then its   also records any accompanying TLP Prefix along with each recorded header. References to header recording also imply TLP Prefix recording.

is recorded, it is recommended that hardware update the First Error Pointer to indicate a status bit that it will never set, e.g., a Reserved status bit. See the Implementation Note below.

If multiple header recording is supported and enabled, and the First Error Pointer is valid, it is recommended that software not write a 1b to any status bit other than the one indicated by the First Error Pointer¹²⁷. If software writes a 1b to such non-indicated bits, hardware is permitted to clear any associated recorded errors, but is not required to do so.

If software observes that the First Error Pointer is invalid, and software wishes to clear any unmasked status bits that were set because of earlier header recording resource overflow, software should be aware of the following race condition. If any new instances of those errors happen to be recorded before software clears those status bits, one or more of the newly recorded errors might be lost.

If multiple header recording is supported and enabled, software must use special care when clearing the Multiple Header Recording Enable bit. Hardware behavior is undefined if software clears that bit while the First Error Pointer is valid. Before clearing the Multiple Header Recording Enable bit, it is recommended that software temporarily mask all uncorrectable errors, and then repetitively dismiss each error indicated by the First Error Pointer.

Since an implementation only has the ability to record a finite number of headers, it is important that software services the First Error Pointer, Header Log, and TLP Prefix Log registers in a timely manner, to limit the risk of missing this information for subsequent errors. A Header Log Overflow occurs when an error that requires header logging is detected and either the number of recorded headers supported by an implementation has been reached, or the Multiple Header Recording Enable bit is not Set and the First Error Pointer is valid.

Implementations may optionally check for this condition and report a Header Log Overflow error. This is a reported error associated with the detecting Function.

The setting of Multiple Header Recording Capable and the checking for Header Log Overflow are independently optional.

127. Status bits for masked errors are an exception. Software can safely clear them if software is certain that they have no recorded headers, as would be the case if they have remained masked since the First Error Pointer was last invalid.

IMPLEMENTATION NOTE: FIRST ERROR POINTER REGISTER BEING VALID §

The First Error Pointer (FEP) field is defined to be valid when the corresponding bit of the Uncorrectable Error Status register is set. To avoid ambiguity with certain cases, the following is recommended:

- After an uncorrectable error has been recorded, when the associated bit in the Uncorrectable Error Status register is cleared by software writing a 1b to it, hardware should update the FEP to point to a status bit that it will never set, e.g., a Reserved status bit. (This assumes that the Function does not already have another recorded error to report, as could be the case if it supports multiple header recording.)
- The default value for the FEP should point to a status bit that hardware will never set, e.g., a Reserved status bit.

Here is an example case of ambiguity with Unsupported Request (UR) if the above recommendations are not followed:

- UR and Advisory Non-Fatal Error are unmasked while system firmware does its Configuration Space probing.
- The Function encounters a UR due to normal probing, logs it, and sets the FEP to point to UR.
- System firmware clears the UR Status bit, and hardware leaves the FEP pointing to UR.
- After the operating system has booted, it masks UR.
- Normal probing sets the UR Status bit, but the error is not recorded since UR is masked.

At this point, there's the ambiguity of the FEP pointing to a status bit that is set (thus being valid), when in fact, there is no recorded error that needs to be processed by software.

If hardware relies on this definition of the FEP being valid to determine when it's possible to record a new error, the Function can fail to record new unmasked errors, falsely determining that it has no available recording resources. Hardware implementations that rely on other internal state to determine when it's possible to record a new error might not have this problem; however, hardware implementations should still follow the above recommendations to avoid presenting this ambiguity to software.

6.2.4.2.1 Multiple Error Handling in VFs §

Header Log space for a PF is independent of that for its associated VFs, and must be implemented with dedicated storage space.

VFs that implement AER may share Header Log space among all the VFs associated with a single PF. Especially when sharing Header Log space, VFs may not have room to log a header associated with an error. In this case, the Function must update the Uncorrectable Error Status Register and Advanced Error Capabilities and Control register as required by § Section 6.2.4.2 ; however, when the Header Log Register is read, it must return all 1s to indicate an overflow condition and that no header was logged. If Flit Mode Supported in the PCI Express Capabilities Register is Set, in addition to returning all 1s, the associated Logged TLP Size field must contain 0.

The VF's header log entry shall be locked and remain valid while that VF's First Error Pointer is valid. As defined in § Section 6.2.4.2 , the First Error Pointer register is valid when the corresponding bit of the Uncorrectable Error Status

register is Set. While the header log entry is locked, additional errors shall not overwrite the locked entry for this or any other VF. When a header entry is unlocked, it shall be available to record a new error for any VF sharing the header logs.

6.2.4.3 Advisory Non-Fatal Error Logging §

§ Section 6.2.3.2.4 describes Advisory Non-Fatal Error cases, under which an agent with AER detecting an uncorrectable error of non-fatal severity signals the error (if enabled) using ERR_COR instead of ERR_NONFATAL. For the same cases, an agent without AER sends no error Message. The remaining discussion in this section is in the context of agents that do implement AER.

For Advisory Non-Fatal Error cases, since an uncorrectable error is signaled using the correctable error Message, control/status/mask bits involving both uncorrectable and correctable errors apply. § Figure 6-2 shows a flowchart of the sequence. Following are some of the unique aspects for logging Advisory Non-Fatal Errors.

First, the uncorrectable error needs to be of severity non-fatal, as determined by the associated bit in the Uncorrectable Error Severity register. If the severity is fatal, the error does not qualify as an Advisory Non-Fatal Error, and will be signaled (if enabled) with ERR_FATAL.

Next, the specific error case needs to be one of the Advisory Non-Fatal Error cases documented in § Section 6.2.3.2.4 . If not, the error does not qualify as an Advisory Non-Fatal Error, and will be signaled (if enabled) with an uncorrectable error Message.

Next, the Advisory Non-Fatal Error Status bit is Set in the Correctable Error Status register to indicate the occurrence of the advisory error, and the Advisory Non-Fatal Error Mask bit in the Correctable Error Mask register is checked, and, if set, no further processing is done.

If the Advisory Non-Fatal Error Mask bit is clear, logging proceeds by setting the “corresponding” bit in the Uncorrectable Error Status register, based upon the specific uncorrectable error that’s being reported as an advisory error. If the “corresponding” uncorrectable error bit in the Uncorrectable Error Mask register is clear and the error is one that requires header logging, then the prefix and header are recorded, subject to the availability of resources. See § Section 6.2.4.2 .

Finally, an ERR_COR Message is sent if the Correctable Error Reporting Enable bit is Set in the Device Control Register .

6.2.4.4 End-End TLP Prefix Logging - Non-Flit Mode §

For any device Function that supports both TLP Prefixes and Advanced Error Reporting the TLP Prefixes associated with the TLP in error are recorded in the TLP Prefix Log register according to the same rules as the Header Log register (such that both the TLP Prefix Log and Header Log registers always correspond to the error indicated in the First Error Pointer, when the First Error Pointer is valid).

The TLP Prefix Log Present bit (see § Section 7.8.4.7) indicates that the TLP Prefix Log register (see § Section 7.8.4.12) contains information.

Only End-End TLP Prefixes are logged by AER. Logging of Local TLP Prefixes may occur elsewhere using prefix-specific mechanisms.

End-End TLP Prefixes are logged in the TLP Prefix Log register. The underlying TLP Header is logged in the Header Log register subject to two exceptions:

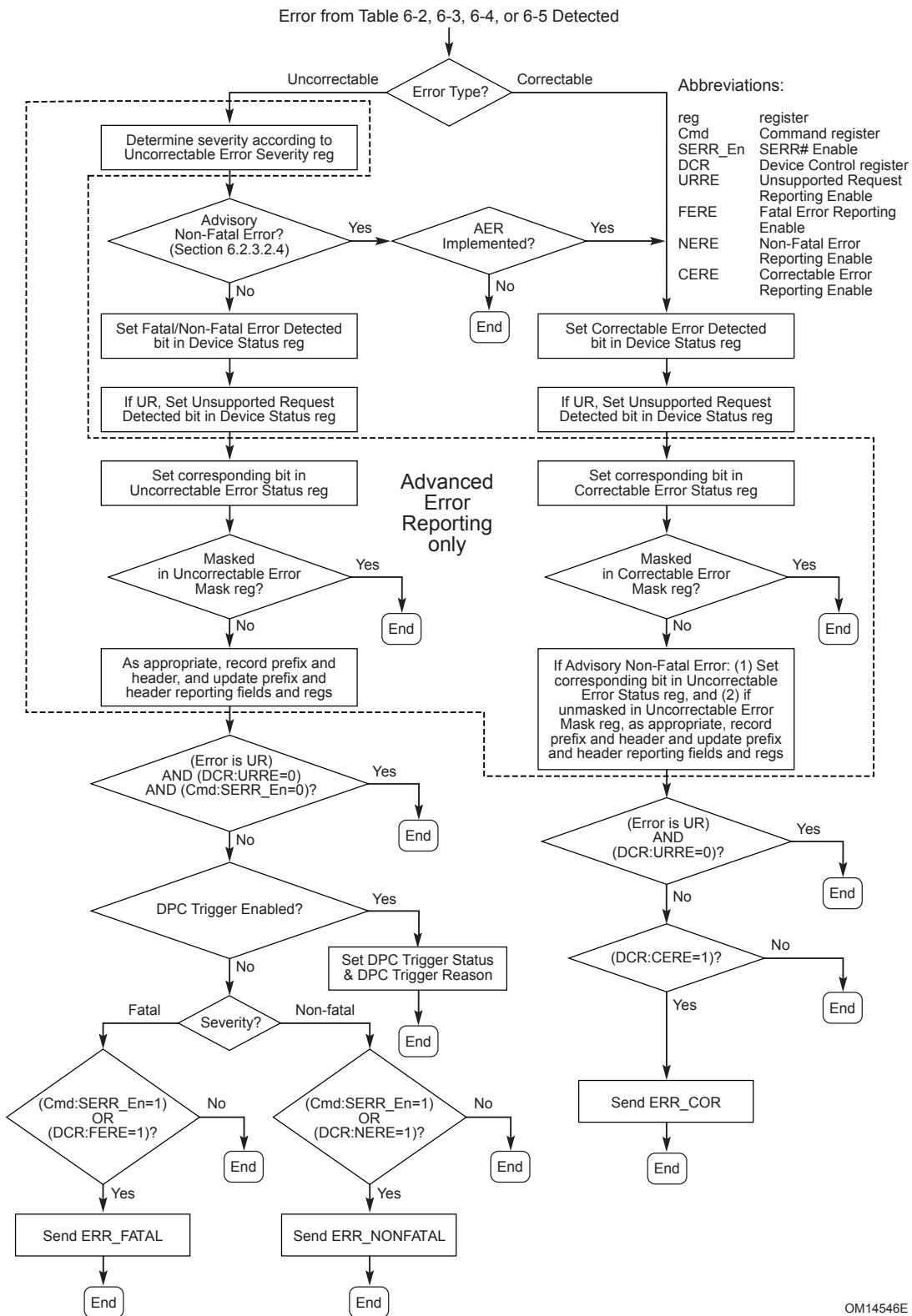
- If the Extended Fmt Field Supported bit is Set (see § Section 7.5.3.15), a Function that does not support End-End TLP Prefixes and receives a TLP containing an End-End TLP Prefix must signal Malformed TLP, and the Header Log register must contain the first four DWs of the TLP (End-End TLP Prefixes followed by as much of the TLP Header as will fit).

- A Function that receives a TLP containing more End-End TLP Prefixes than are indicated by the Function's Max End-End TLP Prefixes field must handle the TLP as an error (see § Section 2.2.10.4 for specifics) and store the first overflow End-End TLP Prefix in the 1st DW of the Header Log register with the remainder of the Header Log register being undefined.

6.2.5 Sequence of Device Error Signaling and Logging Operations §

§ Figure 6-2 shows the sequence of operations related to signaling and logging of errors detected by a device.

Base 6.4 vs Base 6.3



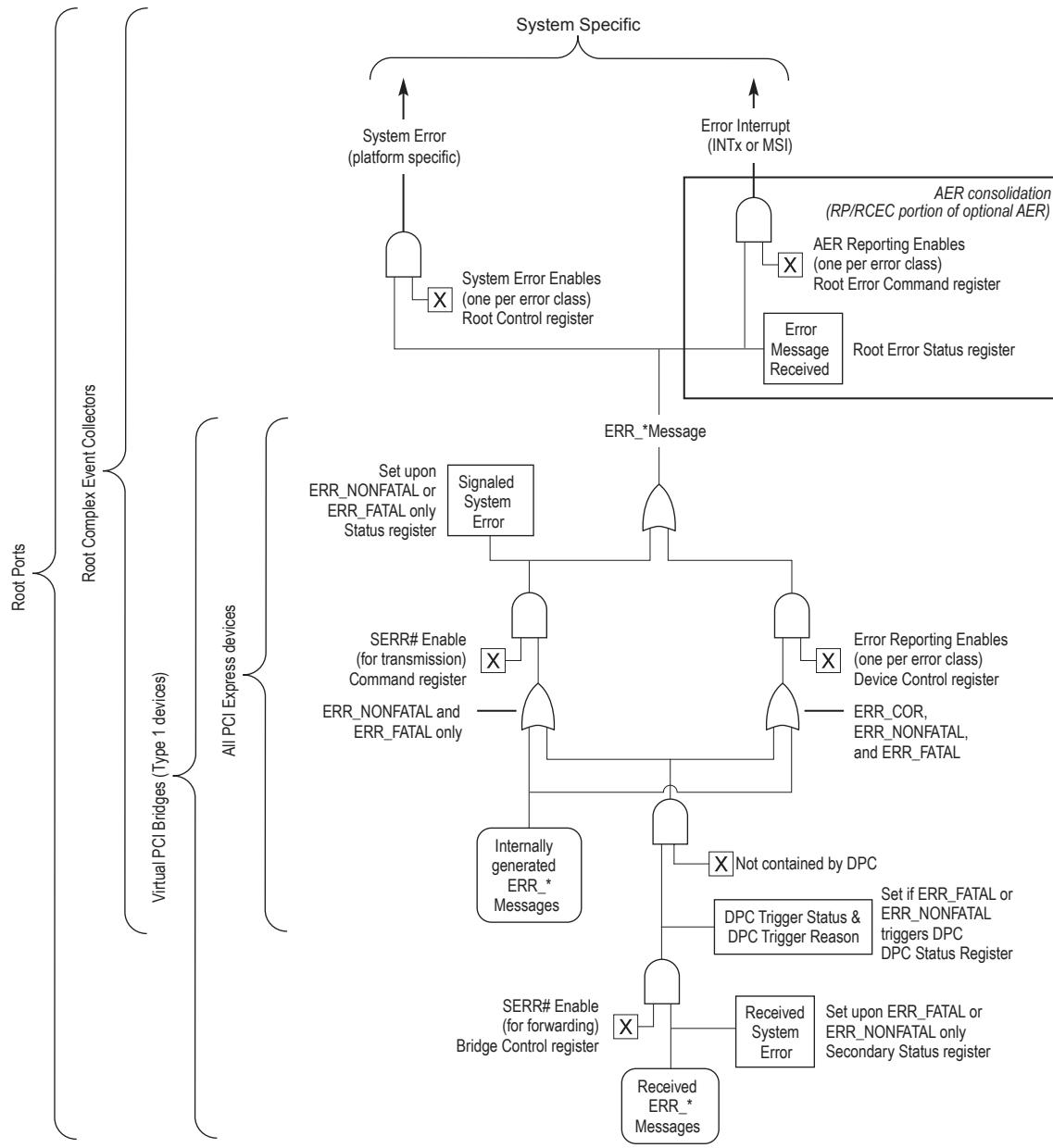
OM14546E

Figure 6-2 Flowchart Showing Sequence of Device Error Signaling and Logging Operations §

Base 6.4 vs Base 6.3

6.2.6 Error Message Controls §

Error Messages have a complex set of associated control and status bits. § Figure 6-3 provides a high-level summary in the form of a pseudo logic diagram for how error Messages are generated, logged, forwarded, and ultimately notified to the system. Not all control and status bits are shown. The logic gates shown in this diagram are intended for conveying general concepts, and not for direct implementation.



A-0479B

Figure 6-3 Pseudo Logic Diagram for Selected Error Message Control and Status Bits §

6.2.7 Error Listing and Rules §

§ Table 6-2 through § Table 6-4 list all of the PCI Express errors that are defined by this specification. Each error is listed with a short-hand name, how the error is detected in hardware, the default severity of the error, and the expected action taken by the agent which detects the error. These actions form the rules for PCI Express error reporting and logging.

The Default Severity column specifies the default severity for the error without any software reprogramming. For device Functions supporting the Advanced Error Reporting Extended Capability, the uncorrectable errors are programmable to Fatal or Non-fatal with the Error Severity register. Device Functions without Advanced Error Reporting Extended Capability use the default associations and are not reprogrammable.

The detecting agent action for Downstream Ports that implement Downstream Port Containment (DPC) and have it enabled will be different if the error triggers DPC. DPC behavior is not described in the following tables. See § Section 6.2.11 for the description of DPC behavior.

Table 6-2 General PCI Express Error List §

| Error Name | Error Type (Default Severity) | Detecting Agent Action ¹²⁸ | References |
|-------------------------------------|--|---|-------------------|
| Corrected Internal Error | Correctable (masked by default) | <i>Component:</i> Send <u>ERR_COR</u> to Root Complex. | § Section 6.2.10 |
| Uncorrectable Internal Error | Uncorrectable (Fatal and masked by default) | <i>Component:</i> Send <u>ERR_FATAL</u> to Root Complex. Optionally, log the prefix/header of the first TLP associated with the error. | § Section 6.2.10 |
| Header Log Overflow | Correctable (masked by default) | <i>Component:</i> Send <u>ERR_COR</u> to Root Complex. | § Section 6.2.4.2 |

Table 6-3 Physical Layer Error List §

| Error Name | Error Type (Default Severity) | Detecting Agent Action ¹²⁹ | References |
|-----------------------|-------------------------------|--|--|
| Receiver Error | Correctable | <i>Receiver:</i> Send <u>ERR_COR</u> to Root Complex. | § Section 4.2.1.1.3 § Section 4.2.1.2 § Section 4.2.5.8 § Section 4.2.7 |

Table 6-4 Data Link Layer Error List §

| Error Name | Error Type (Default Severity) | Detecting Agent Action ¹³⁰ | References |
|-----------------|----------------------------------|--|---|
| Bad TLP | Correctable | <i>Receiver:</i> Send <u>ERR_COR</u> to Root Complex. | ↑↑§ Section 3.6.3.1↑ |
| Bad DLLP | | <i>Receiver:</i> Send <u>ERR_COR</u> to Root Complex. | § Section 3.6.2.2 , and § Section 3.6.2.3 |

128. For these tables, detecting agent action is given as if all enable bits are set to “enable” and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

129. For these tables, detecting agent action is given as if all enable bits are set to “enable” and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

130. For these tables, detecting agent action is given as if all enable bits are set to “enable” and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

| Error Name | Error Type (Default Severity) | Detecting Agent Action | References |
|-------------------------------------|---|---|--|
| Replay Timer Timeout | | <i>Transmitter:</i> Send <u>ERR_COR</u> to Root Complex. | § Section 3.6.2.1 , § Section 4.2.3.4.2.1 |
| REPLAY_NUM Rollover | <i>Transmitter:</i> Send <u>ERR_COR</u> to Root Complex. | § Section 3.6.2.1 , § Section 4.2.3.4.2.1 | |
| Data Link Protocol Error | Uncorrectable (Fatal) | If checking, send <u>ERR_FATAL</u> to Root Complex. | § Section 3.6.2.2, § Section 3.6.2.3, § Section 4.2.3.4.2.1 |
| Surprise Down | | If checking, send <u>ERR_FATAL</u> to Root Complex. | § Section 3.2.1 |

Table 6-5 Transaction Layer Error List [§](#)

| Error Name | Error Type (Default Severity) | Detecting Agent Action ¹³¹ | References |
|--|-------------------------------------|--|-----------------------------------|
| Poisoned TLP Received | Uncorrectable (Non-Fatal) | <i>Receiver:</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error cases described in § Section 6.2.3.2.4.2 and § Section 6.2.3.2.4.3. Log the prefix/header of the Poisoned TLP. ¹³² | § Section 2.7.2.1 |
| Poisoned TLP Egress Blocked | | <i>Downstream Port Transmitter:</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error case described in § Section 6.2.3.2.4.1. Log the prefix/header of the poisoned TLP. | § Section 2.7.2.1 |
| ECRC Check Failed | | <i>Receiver (if ECRC checking is supported):</i> <ul style="list-style-type: none"> • Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error case described in § Section 6.2.3.2.4.1 and § Section 6.2.3.2.4.2. • Log the prefix/header of the TLP that | § Section 2.7.1 |

131. For these tables, detecting agent action is given as if all enable bits are set to “enable” and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

Base 6.4 vs Base 6.3

| Error Name | Error Type (Default Severity) | Detecting Agent Action | References |
|---------------------------------|-------------------------------------|--|---|
| | | encountered the ECRC error. | |
| Unsupported Request (UR) | Uncorrectable (Non-Fatal) | <p><i>Request Receiver:</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error case described in § <u>Section 6.2.3.2.4.1</u>.</p> <p>Log the prefix/header of the TLP that caused the error.</p> | § Table F-1 , § Section 2.3.1 , § Section 2.3.2 , § Section 2.7.2.1 , § Section 2.9.1 , § Section 5.3.1 , § Section 6.2.3.1 , § Section 6.2.6 , § Section 6.2.8.1 , § Section 6.5.7 , § Section 7.3.1 , § Section 7.3.3 , § Section 7.5.1.1.3 , § Section 7.5.1.1.4 |
| Completion Timeout | Uncorrectable (Non-Fatal) | <p><i>Requester:</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error case described in § <u>Section 6.2.3.2.4.4</u>.</p> <p>If the Completion Timeout Prefix/Header Log Capable bit is Set in the Advanced Error Capabilities and Control register, log the prefix/header of the Request TLP that encountered the error.</p> | § Section 2.8 |
| Completer Abort | | <p><i>Completer:</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error case described in § <u>Section 6.2.3.2.4.1</u>.</p> <p>Log the prefix/header of the Request that encountered the error.</p> | § Section 2.3.1 |
| Unexpected Completion | | <p><i>Receiver:</i> Send <u>ERR_COR</u> to Root Complex. This is an Advisory Non-Fatal Error case described in § <u>Section 6.2.3.2.4.5</u>.</p> <p>Log the prefix/header of the Completion that encountered the error.</p> | § Section 2.3.2 |
| ACS Violation | | <p><i>Receiver (if checking):</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the</p> | |

Base 6.4 vs Base 6.3

| Error Name | Error Type (Default Severity) | Detecting Agent Action | References |
|---|-------------------------------------|--|------------------------------------|
| | | <p>Advisory Non-Fatal Error case described in § Section 6.2.3.2.4.1.</p> <p>Log the prefix/header of the Request TLP that encountered the error.</p> | |
| MC Blocked TLP | | <p><i>Receiver (if checking):</i> Send ERR_NONFATAL to Root Complex.</p> <p>Log the prefix/header of the Request TLP that encountered the error.</p> | § Section 6.14.4 |
| AtomicOp Egress Blocked | Uncorrectable (Non-Fatal) | <p><i>Egress Port:</i> Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in § Section 6.2.3.2.4.1.</p> <p>Log the prefix/header of the AtomicOp Request that encountered the error.</p> | § Section 6.15.2 |
| DMWr Request Egress Blocked | | <p><i>Egress Port:</i> Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in § Section 6.2.3.2.4.1.</p> <p>Log the prefix/header of the DMWr Request that encountered the error.</p> | § Section 6.32 |
| TLP Translation Egress Blocked | | <p><i>Egress Port:</i> For error signaling, handle PR/NPR/CPL FC Types as specified in § Section 2.2.1.2.</p> <p>For error logging, handle PR/NPR/CPL FC Types as specified in § Section 2.2.1.2. Log the prefix/header of the Flit Mode TLP that was not possible to translate to send in Non-Flit Mode on the Egress Port.</p> | § Section 2.2.1.2 |
| TLP Prefix Blocked | | <p><i>Egress Port:</i> Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in § Section 6.2.3.2.4.1. Log the prefix/</p> | § Section 2.2.10.4 |

Base 6.4 vs Base 6.3

| Error Name | Error Type (Default Severity) | Detecting Agent Action | References |
|------------------------------------|-------------------------------------|--|---|
| | | header of the TLP that encountered the error. | |
| Receiver Overflow | Uncorrectable (Fatal) | <i>Receiver (if checking):</i> Send <u>ERR_FATAL</u> to Root Complex. | <u>§ Section 2.6.1.2</u> |
| Flow Control Protocol Error | | <i>Receiver (if checking):</i> Send <u>ERR_FATAL</u> to Root Complex. | <u>§ Section 2.6.1</u> |
| Malformed TLP | | <p><i>Receiver:</i> Send <u>ERR_FATAL</u> to Root Complex.</p> <p>Log the prefix/header of the TLP that encountered the error.</p> | <u>§ Section 2.2.2</u> , <u>§ Section 2.2.3</u> , <u>§ Section 2.2.5</u> , <u>§ Section 2.2.7</u> , <u>§ Section 2.2.8.1</u> , <u>§ Section 2.2.8.2</u> , <u>§ Section 2.2.8.3</u> , <u>§ Section 2.2.8.4</u> , <u>§ Section 2.2.8.5</u> , <u>§ Section 2.2.8.10</u> , <u>§ Section 2.2.10</u> , <u>§ Section 2.2.10.2</u> , <u>§ Section 2.2.10.4</u> , <u>§ Section 2.3</u> , <u>§ Section 2.3.1</u> , <u>§ Section 2.3.1.1</u> , <u>§ Section 2.3.2</u> , <u>§ Section 2.5</u> , <u>§ Section 2.5.3</u> , <u>§ Section 2.6.1</u> , <u>§ Section 2.6.1.2</u> , <u>§ Section 6.2.4.4</u> , <u>§ Section 6.3.2</u> |
| IDE Check Failed | | <p><i>Receiving IDE Terminus:</i> Enter Insecure</p> <p>All subsequent IDE TLPs over this stream treated as having MAC check failure</p> <p>Send <u>ERR_FATAL</u> to root complex.</p> <p>Transmit IDE Fail Message to the Partner Port</p> <p>Log the prefix/header of the TLP that encountered the error</p> | <u>§ Section 6.33</u> |
| Misrouted IDE TLP | Uncorrectable (Non-Fatal) | <p><i>Ingress/Egress Port:</i> Send <u>ERR_NONFATAL</u> to Root Complex.</p> <p>Log the prefix/header of the TLP that encountered the error</p> | <u>§ Section 6.33</u> |
| PCRC Check Failed | | <p><i>Receiving IDE Terminus:</i> Send <u>ERR_NONFATAL</u> to Root Complex or <u>ERR_COR</u> for the Advisory Non-Fatal Error cases described in <u>§ Section 6.2.3.2.4.3</u>.</p> <p>Log the prefix/header of the TLP that encountered the error</p> | <u>§ Section 6.33</u> |

For all errors listed above, the appropriate status bit(s) must be set upon detection of the error. For Unsupported Request (UR), additional detection and reporting enable bits apply (see § [Section 6.2.5](#)).

IMPLEMENTATION NOTE: DEVICE UR REPORTING COMPATIBILITY WITH LEGACY AND 1.0A SOFTWARE §

With [PCIe-1.0a] device Functions that do not implement Role-Based Error Reporting¹³³, the Unsupported Request Reporting Enable bit in the Device Control Register, when clear, prevents the Function from sending any error Message to signal a UR error. With Role-Based Error Reporting Functions, if the SERR# Enable bit in the Command Register is set, the Function is implicitly enabled¹³⁴ to send ERR_NONFATAL or ERR_FATAL messages to signal UR errors, even if the Unsupported Request Reporting Enable bit is clear. This raises a backward compatibility concern with software (or firmware) written for [PCIe-1.0a] devices.

With software/firmware that sets the SERR# Enable bit but leaves the Unsupported Request Reporting Enable and Correctable Error Reporting Enable bits clear, a Role-Based Error Reporting Function that encounters a UR error will send no error Message if the Request was non-posted, and will signal the error with ERR_NONFATAL if the Request was posted. The behavior with non-posted Requests supports PC-compatible Configuration Space probing, while the behavior with posted Requests restores error reporting compatibility with PCI and PCI-X, avoiding the potential in this area for silent data corruption. Thus, Role-Based Error Reporting devices are backward compatible with envisioned legacy and [PCIe-1.0a] software and firmware.

6.2.7.1 Conventional PCI Mapping §

In order to support conventional PCI driver and software compatibility, PCI Express error conditions, where appropriate, must be mapped onto the PCI Status register bits for error reporting.

In other words, when certain PCI Express errors are detected, the appropriate PCI Status register bit is Set alerting the error to legacy PCI software. While the PCI Express error results in setting the PCI Status register, clearing the PCI Status register will not result in clearing bits in the Uncorrectable Error Status register and Correctable Error Status register. Similarly, clearing bits in the Uncorrectable Error Status register and Correctable Error Status register will not result in clearing the PCI Status register.

The PCI command register has bits which control PCI error reporting. However, the PCI Command register does not affect the setting of the PCI Express error register bits.

6.2.8 Virtual PCI Bridge Error Handling §

Virtual PCI Bridge configuration headers are associated with each PCI Express Port in a Root Complex or a Switch. For these cases, PCI Express error concepts require appropriate mapping to the PCI error reporting structures.

¹³³. As indicated by the Role-Based Error Reporting bit in the Device Capabilities register . See § [Section 7.8.3](#) .

¹³⁴. Assuming the Unsupported Request Error Mask bit is not set in the Uncorrectable Error Mask Register if the device implements AER.

6.2.8.1 Error Message Forwarding and PCI Mapping for Bridge - Rules §

In general, a TLP is either passed from one side of the Virtual PCI Bridge to the other, or is handled at the ingress side of the Bridge according to the same rules which apply to the ultimate recipient of a TLP. The following rules cover PCI Express specific error related cases. Refer to § Section 6.2.6 for a conceptual summary of Error Message Controls.

- If a Request does not address a space mapped to either the Bridge’s internal space, or to the egress side of the Bridge, the Request is terminated at the ingress side as an Unsupported Request
- Poisoned TLPs are forwarded according to the same rules as non-Poisoned TLPs
 - When forwarding a Poisoned Request Downstream:
 - Set the Detected Parity Error bit in the Status register
 - Set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response Enable bit in the Bridge Control Register is set
 - When forwarding a Poisoned Completion Downstream:
 - Set the Detected Parity Error bit in the Status register
 - Set the Master Data Parity Error bit in the Status register if the Parity Error Response bit in the Command register is set
 - When forwarding a Poisoned Request Upstream:
 - Set the Detected Parity Error bit in the Secondary Status register
 - Set the Master Data Parity Error bit in the Status register if the Parity Error Response bit in the Command register is set
 - When forwarding a Poisoned Completion Upstream:
 - Set the Detected Parity Error bit in the Secondary Status register
 - Set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response Enable bit in the Bridge Control Register is set
- ERR_COR, ERR_NONFATAL, and ERR_FATAL are forwarded from the secondary interface to the primary interface, if the SERR# Enable bit in the Bridge Control Register is set. A Bridge forwarding an error Message must not set the corresponding Error Detected bit in the Device Status register. Transmission of forwarded error Messages by the primary interface is controlled by multiple bits, as shown in § Figure 6-3.
- For a Root Port, error Messages forwarded from the secondary interface to the primary interface must be enabled for “transmission” by the primary interface in order to cause a System Error via the Root Control Register or (when the Advanced Error Reporting Extended Capability is present) reporting via the Root Error Command register and logging in the Root Error Status register and Error Source Identification register.
- For a Root Complex Event Collector (technically not a Bridge), error Messages “received” from associated RCiEPs must be enabled for “transmission” in order to cause a System Error via the Root Control Register or (when the Advanced Error Reporting Extended Capability is present) reporting via the Root Error Command register and logging in the Root Error Status register and Error Source Identification register.

6.2.9 SR-IOV Baseline Error Handling §

All ~~↓↓SR-IOV devices↓~~ ↑↑SR-IOV Devices↑ must support baseline capability, with certain modifications to account for the goal of reduced cost and complexity of implementation.

The following error handling control bits are only implemented in the PF. They are RsvdP in VFs, and VFs must use the control bits in their associated PF for managing their error handling behavior.

- Command register (see § [Section 7.5.1.1.3](#))
 - SERR# Enable
 - Parity Error Response
- Device Control register (see § [Section 7.5.3.4](#))
 - Correctable Reporting Enable
 - Non-Fatal Reporting Enable
 - Fatal Reporting Enable
 - Unsupported Request (UR) Reporting Enable

Each VF must report its error status independently of any other Function. This is necessary to provide SI isolation for errors that are Function-specific. The following baseline error handling status bits must be implemented in each VF:

- Status register (see § [Section 7.5.1.1.4](#))
 - Master Data Parity Error
 - Signaled Target Abort
 - Received Target Abort
 - Received Master Abort
 - Signaled System Error
 - Detected Parity Error
- Device Status register (see § [Section 7.5.3.5](#))
 - Correctable Error Detected
 - Non-Fatal Error Detected
 - Fatal Error Detected
 - Unsupported Request Detected

Each VF must use its own Routing ID when signaling errors.

6.2.10 Internal Errors §

An Internal Error is an error associated with a PCI Express interface that occurs within a component and which may not be attributable to a packet or event on the PCI Express interface itself or on behalf of transactions initiated on PCI Express. The determination of what is considered an Internal Error is implementation specific and is outside the scope of this specification.

Internal Errors may be classified as Corrected Internal Errors or Uncorrectable Internal Errors. A Corrected Internal Error is an error that occurs within a component that has been masked or worked around by hardware without any loss of information or improper operation. An example of a possible Corrected Internal Error is an internal packet buffer memory error corrected by an Error Correcting Code (ECC). An Uncorrectable Internal Error is an error that occurs within a component that results in improper operation of the component. An example of a possible Uncorrectable Internal Error is a memory error that cannot be corrected by an ECC. The only method of recovering from an Uncorrectable Internal Error is reset or hardware replacement.

Reporting of Corrected Internal Errors and Uncorrectable Internal Errors is independently optional. If either is reported, then AER must be implemented.

Header logging is optional for Uncorrectable Internal Errors. When a header is logged, the header is that of the first TLP that was lost or corrupted by the Uncorrectable Internal Error. When header logging is not implemented or a header is not available, a header of all ones is recorded.

Internal Errors that can be associated with a specific PCI Express interface are reported by the Function(s) associated with that Port. Internal Errors detected within Switches that cannot be associated with a specific PCI Express interface are reported by the Upstream Port. Reporting of Internal Errors that cannot be associated with a specific PCI Express interface in all other multi-Port components (e.g., Root Complexes) is outside the scope of this specification.

6.2.11 Downstream Port Containment (DPC) §

Downstream Port Containment (DPC) is an optional normative feature of a Downstream Port. DPC halts PCI Express traffic below a Downstream Port after an unmasked uncorrectable error is detected at or below the Port, avoiding the potential spread of any data corruption, and supporting Containment Error Recovery (CER) if implemented by software. A Downstream Port indicates support for DPC by implementing a DPC Extended Capability structure, which contains all DPC control and status bits. See § Section 7.9.14.

DPC is disabled by default, and cannot be triggered unless enabled by software using the DPC Trigger Enable field. When the DPC Trigger Enable field is set to 01b, DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_FATAL Message. When the DPC Trigger Enable field is set to 10b, DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_NONFATAL or ERR_FATAL Message. In addition to uncorrectable errors of the type managed by the PCI Express Extended Capability and Advanced Error Reporting (AER), RP PIO errors can be handled as uncorrectable errors. See § Section 6.2.11.3. There is also a mechanism described in § Section 6.2.11.4 for software or firmware to trigger DPC.

When DPC is triggered due to receipt of an uncorrectable error Message, the Requester ID from the Message is recorded in the DPC Error Source ID Register and that Message is discarded and not forwarded Upstream. When DPC is triggered by an unmasked uncorrectable error, that error will not be signaled with an uncorrectable error Message, even if otherwise enabled. However, when DPC is triggered, DPC can signal an interrupt or send an ERR_COR Message if enabled. See § Section 6.2.11.1 and § Section 6.2.11.2.

When DPC is triggered, the Downstream Port immediately Sets the DPC Trigger Status bit and DPC Trigger Reason field to indicate the triggering condition (unmasked uncorrectable error, ERR_NONFATAL, ERR_FATAL, RP_PIO error, or software triggered), and disables its Link by directing the LTSSM to the Disabled state. Once the LTSSM reaches the Disabled state, it remains in that state until the DPC Trigger Status bit is Cleared. To ensure that the LTSSM has time to reach the Disabled state or at least to bring the Link down under a variety of error conditions, software must leave the Downstream Port in DPC until the Data Link Layer Link Active bit in the Link Status Register reads 0b; otherwise, the result is undefined. See § Section 7.5.3.8. See § Section 2.9.3 for other important details on Transaction Layer behavior during DPC.

After DPC has been triggered in a Root Port that supports RP Extensions for DPC, the Root Port may require some time to quiesce and clean up its internal activities, such as those associated with DMA read Requests. When the DPC Trigger Status bit is Set and the DPC RP Busy bit is Set, software must leave the Root Port in DPC until the DPC RP Busy bit reads 0b.

After software releases the Downstream Port from DPC, the Port's LTSSM must transition to the Detect ↑↓state,↓
↑↑state unless the Link is disabled due to other conditions (e.g., the Link Disable bit being Set),↑
where the Link will attempt to retrain. Software can use Data Link Layer State Changed interrupts, DL_ACTIVE ERR_COR signaling, or both, to signal when the Link reaches the DL_Active state again.
See § Section 6.7.3.3 and § Section 6.2.11.5.

Errata: Base 6.3
B828△◀▷

IMPLEMENTATION NOTE: DATA VALUE OF ALL 1'S §

Many platforms, including those supporting RP Extensions for DPC, can return a data value of all 1's to software when an error is associated with a PCI Express Configuration, I/O, or Memory Read Request. During DPC, the Downstream Port discards Requests destined for the Link and completes them with an error (i.e., either with an Unsupported Request (UR) or Completer Abort (CA) Completion Status). By ending a series of MMIO or configuration space operations with a read to an address with a known data value not equal to all 1's, software may determine if a Completer has been removed or DPC has been triggered.

Also see the Implementation Note “ [Use of RP PIO Advisory Error Handling](#) ”

IMPLEMENTATION NOTE: SELECTING A RESPONSE FOR A NON-POSTED REQUEST OR A UIO REQUEST RESPONSE DURING DPC §

The DPC Completion Control bit determines how a Downstream Port responds to a Non-Posted Request (NPR) [↑↑or a UIO Request↑](#) received during DPC. The selection needs to take into account how the rest of the platform handles Containment Error Recovery (CER).

Errata: Base 6.3
B834△◀▶

While specific CER policy details in a platform are outside the scope of this specification, here are some guidelines based on general considerations.

If the platform or drivers do not support CER policies, it's recommended to select UR Completions, which is the standard behavior when a device is not present.

If the CER strategy relies on software detecting containment by looking for all 1's returned by PIO reads, then a UR Completion may be the more appropriate selection, assuming the RP synthesizes an all 1's return value for PIO reads that return UR Completions. The all 1's synthesis would need to occur for PIO reads that target Configuration Space, Memory Space, and perhaps I/O Space.

If the CER strategy utilizes a mechanism that handles UR and CA Completions differently for PIO reads, then a CA Completion might be the more appropriate selection. CA Completions coming back from a PCIe device normally indicate a device programming model violation, which may need to trigger Port containment and error recovery.

IMPLEMENTATION NOTE: SELECTING THE DPC TRIGGER CONDITION §

Non-Fatal Errors are uncorrectable errors that indicate that a particular TLP was unreliable, and in general the associated Function should not continue its normal operation. Fatal errors are uncorrectable errors that indicate that a particular Link and its related hardware are unreliable, and in general the entire hierarchy below that Link should not continue normal operation. This distinction between Non-Fatal and Fatal errors together with the Root Port error containment capabilities can sometimes be used to select the appropriate DPC trigger condition. The following assumes that there is no peer-to-peer traffic between devices.

Some RCs implement a proprietary feature that will be referred to generically as “Function Level Containment” (FLC). This is not an architected feature of PCI Express. A Root Port that implements FLC is capable of containing the traffic associated with a specific Function when a Non-Fatal Error is detected in that traffic. Switch Downstream Ports below a Root Port with FLC should be configured to trigger DPC when the Downstream Port detects an unmasked uncorrectable error itself or when the Downstream Port receives an ERR_FATAL Message. Under this mode, the Switch Downstream Port passes ERR_NONFATAL Messages it receives Upstream without triggering DPC. This enables Root Port FLC to handle Non-Fatal Errors that render a specific Function unreliable and Switch Downstream Port DPC to handle errors that render a subtree of the hierarchy domain unreliable. The Downstream Port still needs to trigger DPC for all unmasked uncorrectable errors it detects, since an ERR_NONFATAL it generates will have its own Requester ID, and the FLC hardware in the Root Port would not be able to determine which specific Function below the Switch Downstream Port was responsible for the Non-Fatal Error.

Switch Downstream Ports below a Root Port without FLC should be configured to trigger DPC when the Switch Downstream Port detects an unmasked uncorrectable error or when the Switch Downstream Port receives an ERR_NONFATAL or ERR_FATAL Message. This enables DPC to contain the error to the affected hierarchy below the Link and allow continued normal operation of the unaffected portion of the hierarchy domain.

IMPLEMENTATION NOTE: SOFTWARE POLLING THE DPC RP BUSY BIT §

The DPC RP Busy bit is a means for hardware to indicate to software that the RP needs to remain in DPC containment while the RP does some internal cleanup and quiescing activities. While the details of these activities are implementation specific, the activities will typically complete within a few microseconds or less. However, under worst-case conditions such as those that might occur with certain internal errors in large systems, the busy period might extend substantially, possibly into multiple seconds. If software is unable to tolerate such lengthy delays within the current software context, software may need to rely on using timer interrupts to schedule polling under interrupt.

IMPLEMENTATION NOTE: DETERMINATION OF DPC CONTROL §

DPC may be controlled in some configurations by platform firmware and in other configurations by the operating system. DPC functionality is strongly linked with the functionality in Advanced Error Reporting. To avoid conflicts over whether platform firmware or the operating system have control of DPC, it is recommended that platform firmware and operating systems always link the control of DPC to the control of Advanced Error Reporting.

6.2.11.1 DPC Interrupts §

A DPC-capable Downstream Port must support the generation of DPC interrupts. DPC interrupts are enabled by the DPC Interrupt Enable bit in the DPC Control Register. DPC interrupts are indicated by the DPC Interrupt Status bit in the DPC Status Register.

If the Port is enabled for level-triggered interrupt signaling using INTx messages, the virtual INTx wire must be asserted whenever and as long as the following conditions are satisfied:

- The value of the Interrupt Disable bit in the Command register is 0b.
- The value of the DPC Interrupt Enable bit is 1b.
- The value of the DPC Interrupt Status bit is 1b.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If the Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- The value of the DPC Interrupt Enable bit is 1b.
- The value of the DPC Interrupt Status bit is 1b.

The Port may optionally send an interrupt message if interrupt generation has been disabled, and the logical AND of the above conditions is TRUE when interrupt generation is subsequently enabled.

The interrupt message will use the vector indicated by the DPC Interrupt Message Number field in the DPC Capability register. This vector may be the same or may be different from the vectors used by other interrupt sources within this Function.

6.2.11.2 DPC ERR_COR Signaling §

A DPC-capable Downstream Port must support ERR_COR signaling, independent of whether it supports Advanced Error Reporting (AER) or not. DPC ERR_COR signaling is enabled by the DPC ERR_COR Enable bit in the DPC Control Register. DPC triggering is indicated by the DPC Trigger Status bit in the DPC Status Register. DPC ERR_COR signaling is managed independently of DPC interrupts, and it is permitted to use both mechanisms concurrently.

If the DPC ERR_COR Enable bit is Set, and the Correctable Error Reporting Enable bit in the Device Control Register or the DPC SIG_SFW Enable bit in the DPC Control Register is Set, the Port must send an ERR_COR Message each time the DPC Trigger Status bit transitions from Clear to Set. DPC ERR_COR signaling must not Set the Correctable Error Detected bit in

the Device Status Register , since this event is not handled as an error. If the Downstream Port supports ERR_COR Subclass capability, this DPC ERR_COR signaling event must set the DPC SIG_SFW Status bit in the DPC Status Register and also set the ERR_COR Subclass field in the ERR_COR Message to indicate ECS SIG_SFW .

For a given DPC trigger event, if a Port is going to send both an ERR_COR Message and an MSI/MSI-X transaction, then the Port must send the ERR_COR Message prior to sending the MSI/MSI-X transaction. There is no corresponding requirement if the INTx mechanism is being used to signal DPC interrupts, since INTx Messages won't necessarily remain ordered with respect to ERR_COR Messages when passing through routing elements.

IMPLEMENTATION NOTE: USE OF DPC ERR_COR SIGNALING §

It is recommended that operating systems use DPC interrupts for signaling when DPC has been triggered. While DPC ERR_COR signaling indicates the same event, DPC ERR_COR signaling is primarily intended for use by system firmware, when it needs to be notified in order to do its own logging of the event or provide firmware first services.

6.2.11.3 Root Port Programmed I/O (RP PIO) Error Controls §

The RP PIO error control registers enable fine-grained control over what happens when Non-Posted Requests ↑↑and UIO Requests↑ that are tracked by the Root Port encounter certain uncorrectable or advisory errors. See § Section 2.9.3 for a description of which Non-Posted Requests are tracked. A set of control and status bits exists for receiving Completion with Unsupported Request status (UR Cpl), receiving Completion with Completer Abort status (CA Cpl), and Completion Timeout (CTO) errors. Independent sets of these error bits exist for Configuration Requests, I/O Requests, and Memory Requests. This finer granularity enables more precise error handling for this subset of uncorrectable errors (UR Cpl, CA Cpl, and CTO). As a key example, UR Cpl errors with Memory Read Requests can be configured to trigger DPC for proper containment and error handling, while UR Cpl errors with Configuration Requests can be configured to return all 1's (without triggering DPC) for normal probing and enumeration.

Errata: Base 6.3
B834△◀▷

A UR or CA error logged in AER is the result of the Root Port operating in the role of a Completer ↑↓and, ↓↑returning a Completion↑ for a ↓↓received↓ Non-Posted ↓↓Request, returning a Completion.↑ ↑↑Request or UIO Request.↑ In contrast, a UR Cpl or CA Cpl error logged as an RP PIO error is the result of the Root Port operating in the role of a Requester, and ↑↑receiving a Completion↑ for ↑↓an outstanding↓ ↑↑a↑ Non-Posted ↓↓Request, receiving↓ ↑↑Request or↑ a ↓↓Completion.↓ ↑↑UIO Request.↑ CTO errors logged in both AER and RP PIO are the result of the Root Port operating in the role of a Requester, though the RP PIO error controls support per-space granularity. Depending upon the control register settings, CTO errors can be logged in AER registers, in RP PIO registers, or both. If software unmasks CTO errors in RP PIO, it is recommended that software mask CTO errors in AER ↓↓in order↓ to avoid unintended interactions.

Errata: Base 6.3
B834△◀▷

The RP PIO Header Log Register , RP PIO ImpSpec Log Register , and RP PIO TLP Prefix Log Registers are referred to collectively as the RP PIO log registers. The RP PIO Header Log Register must be implemented; the RP PIO ImpSpec Log Register and RP PIO TLP Prefix Log Register are optional. The RP PIO Log Size field indicates how many DWORDs are allocated for the RP PIO log registers, and from this the allocated size for the RP PIO TLP Prefix Log Register can be calculated. See § Section 7.9.14.2 . The RP PIO log registers always record information from a PIO Request, not any associated Completions.

When Flit Mode Supported is Set and the link is operating in Flit Mode, the RP PIO Header Log Register extends into additional DWs as indicated by the RP PIO Log Size field. Software must parse the Type and OHC fields to determine the size and layout of a TLP recorded in the RP PIO Header Log Register . Hardware is not required to support logging of TLP Headers larger than the largest size supported by the Port. Hardware is not required to support logging of OHC types not

supported by the Port. TLP Trailers are not logged in the RP PIO Header Log Register . The required minimum size of the RP PIO Header Log Register is determined by the largest Header Base Size implemented by the Port (up to the maximum defined of 7 DW - See § Table 2-5), plus the largest number of OHC implemented by the Port (up to the maximum defined of 7 DW). Hardware must hardwire to zero the DW of the RP PIO Header Log Register beyond those required to log the largest supported TLP Header and the overall length of the Advanced Error Reporting Extended Capability is reduced accordingly. As in Non-Flit Mode, Local TLP Prefixes are not logged.

The RP PIO Status, Mask, and Severity registers behave similarly to the Uncorrectable Error Status, Mask, and Severity registers in AER. See § Section 7.8.4.2 , § Section 7.8.4.3 , and § Section 7.8.4.4 . When an RP PIO error is detected while it is unmasked, the associated bit in the RP PIO Status Register is Set, and the error is recorded in the RP PIO log registers (assuming that RP PIO error logging resources are available). When an RP PIO error is detected while it is masked, the associated bit is still Set in the RP PIO Status Register , but the error does not trigger DPC and the error is not recorded in the RP PIO log registers.

Each unmasked RP PIO error is handled either as uncorrectable or advisory, as determined by the value of the corresponding bit in the RP PIO Severity Register . If the associated Severity bit is Set, the error is handled as uncorrectable, triggering DPC (assuming that DPC is enabled) and signaling this event with a DPC interrupt and/or ERR_COR (if enabled). If the associated Severity bit is Clear, the error is handled as advisory (without triggering DPC) and signaled with ERR_COR (if enabled).

IMPLEMENTATION NOTE: USE OF RP PIO ADVISORY ERROR HANDLING §

Each RP PIO error can be handled either as uncorrectable or advisory. Uncorrectable error handling usually logs the error, triggers DPC, and signals the event either with a DPC interrupt, an ERR_COR, or both. Advisory error handling usually logs the error and signals the event with ERR_COR.

RP PIO advisory error handling can be used by software in certain cases to handle RP PIO errors robustly without incurring the disruption caused if DPC is triggered in the RP. If an RP PIO Exception is not enabled for a given error, an all 1's value is returned whenever the error occurs. If the error does not trigger DPC, software may be uncertain if the all 1's value returned by a given PIO read is the actual data value returned by the Completion versus indicating that an error occurred with that PIO read. If software enables advisory error handling for that error, instances of that error will be logged, enabling software to distinguish the two cases.

The use of RP PIO advisory error handling is notably beneficial if DPC is triggered in a Switch Downstream Port, and that causes one or more Completion Timeouts in the RP as a side-effect, as described in § Section 2.9.3 . If the RP handles Completion Timeout errors as advisory, this avoids DPC being triggered in the RP, permitting continued operation with the other Switch Downstream Ports.

The RP PIO First Error Pointer, RP PIO Header Log, and RP PIO TLP Prefix Log behave similarly to the First Error Pointer, Header Log, and TLP Prefix Log in AER. The RP PIO First Error Pointer is defined to be valid when its value indicates a bit in the RP PIO Status Register that is Set. When the RP PIO First Error Pointer is valid, the RP PIO log registers contain the information associated with the indicated error. The RP PIO ImpSpec Log, if implemented, contains implementation specific information, e.g., the source of the Request TLP.

In contrast to AER, where the recording of CTO error information in the AER log registers is optional, RP PIO implementations must support recording RP PIO CTO error information in the RP PIO log registers.

If an error is detected with a received Completion TLP associated with an outstanding PIO Request, the set of RP PIO error control bits used to govern the error handling is determined in a similar manner. The DPC Completion Control bit determines whether UR or CA applies, and the Space (Configuration, I/O, or Memory) is that of the associated PIO Request. For example, if the DPC Completion Control bit is configured for CA, and a Root Port receives a poisoned

Completion for a PIO Memory Read Request, the Mem CA Cpl bit (bit 17) is used in the RP PIO control and status registers for handling the error.

The RP PIO SysError Register provides a means to generate a System Error when an RP PIO error occurs. If an unmasked RP PIO error is detected while its associated bit in the RP PIO SysError Register is Set, a System Error is generated.

The RP PIO Exception Register provides a means to generate a synchronous processor exception¹³⁵ when an error occurs with certain tracked Non-Posted Requests **↑↑and UIO Requests↑** that are generated by a processor instruction. See § Section 2.9.3 . This exception must support all such tracked **↑↓read↓** Requests, and may optionally support Configuration write, I/O write, and AtomicOp Requests. If an error with an exception-supported Non-Posted Request **↑↑or UIO Request↑** is detected¹³⁶ or a Completion for it is synthesized, and its associated bit in the RP PIO Exception Register is Set, the processor instruction that generated the **↑↓Non-Posted↓** Request must take a synchronous exception. This still applies even if the RP PIO or AER controls specify that the error be handled as masked or advisory.

Errata: Base 6.3
B834△◀▷

The details of a processor instruction taking a synchronous exception are processor-specific, but at a minimum, the mechanism must be able to interrupt the normal processor instruction flow either before completion of the instruction that generated the Non-Posted **↑↑Request or UIO↑** Request, or immediately following that instruction. The intent is that exception handling routines in system firmware, the operating system, or both, can examine the cause of the exception and take corrective action if necessary.

Errata: Base 6.3
B834△◀▷

If an RP PIO error occurs with a processor-generated read or AtomicOp Request, and the RP PIO Exception Register value does not cause an exception, a value of all 1's must be returned for the instruction that generated the Request.

IMPLEMENTATION NOTE: SYNCHRONOUS EXCEPTION IMPLEMENTATION §

The exact mechanism for implementing synchronous exceptions is processor and platform specific. One possible implementation is poisoning the data returned to the processor for a read or AtomicOp Request that encounters an error. While this approach is likely to work with those Requests, it might not work with Configuration and I/O write Requests since they return no data.

Another possible implementation is marking the response transaction for processor-generated Non-Posted Requests with some other type of indication of the Request having failed, e.g., a “hard fail” response. This approach is more likely to work with all processor-generated Non-Posted Requests.

135. “Exception” is used as a generic term for a variety of mechanisms used by processors, including interrupts, traps, machine checks, instruction aborts, etc.

136. This includes any errors with the Completion TLP itself (e.g., Malformed TLP) or where the Completion Status is other than Successful Completion.

IMPLEMENTATION NOTE: RP PIO MASK BIT BEHAVIOR AND RATIONALE §

For a given RP PIO error, the associated mask bit in the RP PIO Mask Register affects its associated status bit setting, error logging, and error signaling in a manner that closely parallels the behavior of mask bits in AER.

SysError generation for a given RP PIO error is primarily controlled by the associated bit in the RP PIO SysError Register, but is also contingent upon the associated RP PIO mask bit being Clear. This behavior was chosen for consistency with AER, and also since it is poor practice to generate a SysError without logging the reason.

Exception generation for a given RP PIO error is independent of the associated RP PIO mask bit value. Usage
~~↓↓Models↓~~ ↑↑models↑ are envisioned where an RP PIO error needs to generate an Exception without logging an RP PIO error or triggering DPC.

Root Port error handling for tracked Non-Posted Requests ↑↑and UIO Requests↑ with errors other than receiving UR and CA Completions is governed by a combination of AER and RP PIO error controls. Examples are CTO¹³⁷, Poisoned TLP Received, and Malformed TLP. For a given error managed by AER, the associated AER Mask and Severity bits determine if the error must be handled as an uncorrectable error, handled as an Advisory Non-Fatal Error, or handled as a masked error.

Errata: Base 6.3
B834△◀▷

- If the AER-managed error is to be handled as an uncorrectable error (see § Section 6.2.2.2), DPC is triggered. The RP PIO SysError and RP PIO Exception bits associated with the Request type and Completion Status apply.
- If the AER-managed error is to be handled as an Advisory Non-Fatal Error (see § Section 6.2.3.2.4), DPC is not triggered. The RP PIO SysError and RP PIO Exception bits do apply.
- If the AER-managed error is to be handled as a masked error (see § Section 6.2.3.2.2), DPC is not triggered. RP PIO SysError bit does not apply, but the RP PIO Exception bit does apply.

6.2.11.4 Software Triggering of DPC §

If the DPC Software Triggering Supported bit in the DPC Capability register is Set, then software can trigger DPC by writing a 1b to the DPC Software Trigger bit in the DPC Control Register, assuming that DPC is enabled and the Port isn't currently in DPC. This mechanism is envisioned to be useful for software and/or firmware development and testing. It also supports usage models where software or firmware examines RP PIO Exceptions or RP PIO advisory errors, and decides to trigger DPC based upon the situation.

When this mechanism triggers DPC, the DPC Trigger Reason and DPC Trigger Reason Extension fields in the DPC Status Register will indicate this as the reason.

If a Port is already in DPC when a 1b is written to the DPC Software Trigger bit, the Port remains in DPC, and the DPC Trigger Reason and DPC Trigger Reason Extension fields are not modified.

¹³⁷. CTO errors have status and mask bits in both AER and RP PIO, though RP PIO has independent sets of bits for each of the 3 spaces. Other errors in AER have no equivalent errors in RP PIO.

IMPLEMENTATION NOTE: AVOID DISABLE LINK AND HOT-PLUG SURPRISE USE WITH DPC §

It is recommended that software not Set the Link Disable bit in the Link Control register while DPC is enabled but not triggered. Setting the Link Disable bit will cause the Link to be directed to DL_Down, invoking some semantics similar to those in DPC, but lacking others. If DPC is enabled, the subsequent arrival of any Posted Requests will likely trigger DPC anyway. If DPC is enabled, the recommended method for software to disable the Link is to write a 1b to the optional DPC Software Trigger bit in the DPC Control Register. If the DPC Software Trigger bit is not implemented, software should disable DPC and use Link Disable instead. If the operating system is performing this action, but DPC is owned by system firmware, the operating system should coordinate disabling DPC with system firmware.

DPC is not recommended for use concurrently with the Hot-Plug Surprise mechanism, indicated by the Hot-Plug Surprise bit in the Slot Capabilities register being Set. Having this bit Set blocks the reporting of Surprise Down errors, preventing DPC from being triggered by this important error, greatly reducing the benefit of DPC. See § Section 6.7.4.5 for guidance on slots supporting both mechanisms.

6.2.11.5 DL_Active ERR_COR Signaling §

Support for this feature is indicated by the DL_Active ERR_COR Signaling Supported bit in the DPC Capability register. The feature is enabled by the DL_ACTIVE ERR_COR Enable bit in the DPC Control Register. The DL_ACTIVE state is indicated by the Data Link Layer Link Active bit in the Link Status Register. DL_ACTIVE ERR_COR signaling is managed independently of Data Link Layer State Changed interrupts, and it is permitted to use both mechanisms concurrently.

If the DL_ACTIVE ERR_COR Enable bit is Set, and the Correctable Error Reporting Enable bit in the Device Control register or the DPC SIG_SFW Enable bit in the DPC Control Register is Set, the Port must send an ERR_COR Message each time the Link transitions into the DL_Active state. DL_ACTIVE ERR_COR signaling must not Set the **↓↑Correctable Error Detected↓↑Correctable Error Detected↑** bit in the **↓↑Device Status register,↓↑Device Status register,↑** since this event is not handled as an error. If the Downstream Port supports ERR_COR Subclass capability, this DPC ERR_COR signaling event must set the DPC SIG_SFW Status bit in the DPC Status register and also set the ERR_COR Subclass field in the ERR_COR Message to indicate ECS SIG_SFW. In contrast to **↓↑Data Link Layer State Changed↓↑Data Link Layer State Changed↑** interrupts, DL_ACTIVE ERR_COR signaling only indicates the Link enters the DL_Active state, not when the Link exits the DL_Active state.

For a given DL_ACTIVE event, if a Port is going to send both an ERR_COR Message and an MSI/MSI-X transaction, then the Port must send the ERR_COR Message prior to sending the MSI/MSI-X transaction. There is no corresponding requirement if the INTx mechanism is being used to signal DL_ACTIVE interrupts, since INTx Messages won't necessarily remain ordered with respect to ERR_COR Messages when passing through routing elements.

IMPLEMENTATION NOTE: USE OF DL_ACTIVE ERR_COR SIGNALING §

It is recommended that operating systems use Data Link Layer State Changed interrupts for signaling when DL_ACTIVE changes state. While DL_ACTIVE ERR_COR signaling indicates a subset of the same events, DL_ACTIVE ERR_COR signaling is primarily intended for use by system firmware, when it needs to be notified in order to do Downstream Port configuration or provide firmware first services.

6.3 Virtual Channel Support §

6.3.1 Introduction and Scope §

The Virtual Channel mechanism provides a foundation for supporting differentiated services within the PCI Express fabric. It enables deployment of independent physical resources that together with traffic labeling are required for optimized handling of differentiated traffic. Traffic labeling is supported using Traffic Class TLP-level labels. The policy for traffic differentiation is determined by the TC/VC mapping and by the VC-based, Port-based, and Function-based arbitration mechanisms supported by certain VC capabilities. The TC/VC mapping depends on the platform application requirements. These requirements drive the choice of the arbitration algorithms and configurability/programmability of arbiters allows detailed tuning of the traffic servicing policy.

The definition of the Virtual Channel and associated Traffic Class mechanisms is covered in § Chapter 2.. The VC configuration/programming models are defined in § Section 7.9.1 , § Section 7.9.2 , and § Section 7.9.29 .

This section covers VC mechanisms from the system perspective. It addresses the next level of details on:

- Supported TC/VC configurations
- VC-based arbitration - algorithms and rules
- Traffic ordering considerations
- Isochronous support as a specific usage model
- SVC and VC/MFVC capability coexistence

6.3.2 TC/VC Mapping and Example Usage §

A Virtual Channel is established when one or more TCs are associated with a physical resource designated by a VC ID. Every Traffic Class that is supported on a given path within the fabric must be mapped to one of the enabled Virtual Channels. Every Port must support the default TC0/VC0 pair - this is “hardwired”. Any additional TC mapping or additional VC resource enablement is optional and is controlled by system software using the programming model described in Sections 7.9.1 and 7.9.2.

The number of VC resources provisioned within a component or enabled within a given fabric may vary due to implementation and usage model requirements, due to Hot-Plug of disparate components with varying resource capabilities, or due to system software restricting what resources may be enabled on a given path within the fabric.

Some examples to illustrate:

Base 6.4 vs Base 6.3

- A set of components (Root Complex, Endpoints, Switches) may only support the mandatory VC0 resource that must have TC0 mapped to VC0. System software may, based on application usage requirements, map one or all non-zero TCs to VC0 as well on any or all paths within the fabric.
- A set of components may support two VC resources, e.g., VC0 and VC1. System software must map TC0/VC0 and in addition, may map one or all non-zero TC labels to either VC0 or VC1. As above, these mappings may be enabled on any or all paths within the fabric. Refer to the examples below for additional information.
- A Switch may be implemented with eight Ports - seven x1 Links with two VC resources and one x16 Link with one VC resource. System software may enable both VC resources on the x1 Links and assign one or more additional TCs to either VC thus allowing the Switch to differentiate traffic flowing between any Ports. The x16 Link must also be configured to map any non-TC0 traffic to VC0 if such traffic is to flow on this Link. Note: multi-Port components (Switches and Root Complex) are required to support independent TC/VC mapping per Port.

In any of the above examples, system software has the ability to map one, all, or a subset of the TCs to a given VC. Should system software wish to restrict the number of traffic classes that may flow through a given Link, it may configure only a subset of the TCs to the enabled VC resources. Any TLP indicating a TC that has not been mapped to an enabled VC resource must be treated as a Malformed TLP. This is referred to as TC Filtering. Flow Control credits for this TLP will be lost, and an uncorrectable error will be generated, so software intervention will usually be required to restore proper operation after a TC Filtering event occurs.

A graphical example of TC filtering is illustrated in § Figure 6-4 , where TCs (2:6) are not mapped to the Link that connects Endpoint A and the Switch. This means that the TLPs with TCs (2:6) are not allowed between the Switch and Endpoint A.

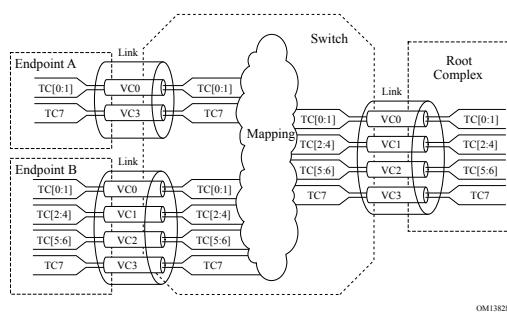


Figure 6-4 TC Filtering Example §

§ Figure 6-5 shows an example of TC to VC mapping. A simple Switch with one Downstream Port and one Upstream Port connects an Endpoint to a Root Complex. At the Upstream Port, two VCs (VC0 and VC1) are enabled with the following mapping: TC(0-6)/VC0, TC7/VC1. At the Downstream Port, only VC0 is enabled and all TCs are mapped to VC0. In this example while TC7 is mapped to VC0 at the Downstream Port, it is re-mapped to VC1 at the Upstream Port. Although the Endpoint only supports VC0, when it labels transactions with different TCs, transactions associated with TC7 from/to the Endpoint can take advantage of the second Virtual Channel enabled between the Switch and the Root Complex.

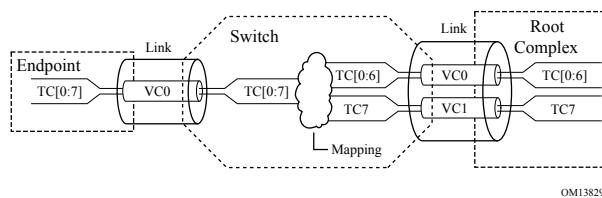


Figure 6-5 TC to VC Mapping Example §

IMPLEMENTATION NOTE: MULTIPLE TCS OVER A SINGLE VC §

A single VC implementation may benefit from using multiple TCs. TCs provide ordering domains that may be used to differentiate traffic within the Endpoint or the Root Complex independent of the number of VCs supported.

In a simple configuration, where only VC0 is supported, traffic differentiation may not be accomplished in an optimum manner since the different TCs cannot be physically segregated. However, the benefits of carrying multiple TCs can still be exploited particularly in the small and “shallow” topologies where Endpoints are connected directly to Root Complex rather than through cascaded Switches. In these topologies traffic that is targeting Root Complex only needs to traverse a single Link, and an optimized scheduling of packets on both sides (Endpoint and Root Complex) based on TCs may accomplish significant improvement over the case when a single TC is used. Still, the inability to route differentiated traffic through separate resources with fully independent flow control and independent ordering exposes all of the traffic to the potential head-of-line blocking conditions. Optimizing Endpoint internal architecture to minimize the exposure to the blocking conditions can reduce those risks.

6.3.3 VC Arbitration §

Arbitration is one of the key aspects of the Virtual Channel mechanism and is defined in a manner that fully enables configurability to the specific application. In general, the definition of the VC-based arbitration mechanism is driven by the following objectives:

- To prevent false transaction timeouts and to guarantee data flow forward progress
- To provide differentiated services between data flows within the fabric
- To provide guaranteed bandwidth with deterministic (and reasonably small) end-to-end latency between components

Links are bidirectional, i.e., each Port can be an Ingress or an Egress Port depending on the direction of traffic flow. This is illustrated by the example of a 3-Port Switch in § Figure 6-6 , where the paths for traffic flowing between Switch Ports are highlighted with different types of lines. In the following sections, VC Arbitration is defined using a Switch arbitration model since the Switch represents a functional superset from the arbitration perspective.

In addition, one-directional data flow is used in the description.

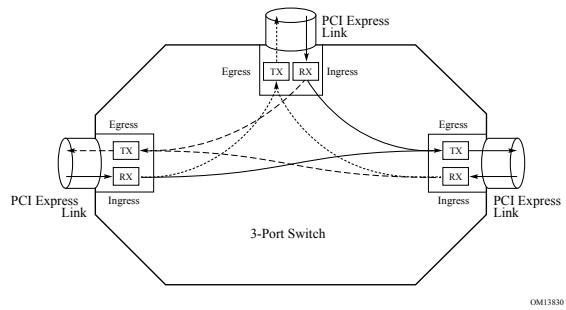


Figure 6-6 An Example of Traffic Flow Illustrating Ingress and Egress §

6.3.3.1 Traffic Flow and Switch Arbitration Model §

The following set of figures (§ Figure 6-7 and § Figure 6-8) illustrates traffic flow through the Switch and summarizes the key aspects of the arbitration.

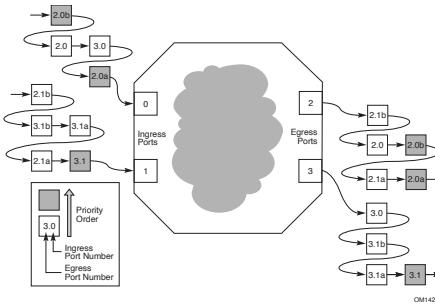


Figure 6-7 An Example of Differentiated Traffic Flow Through a Switch §

At each Ingress Port an incoming traffic stream is represented in § Figure 6-7 by small boxes. These boxes represent packets that are carried within different VCs that are distinguished using different levels of gray. Each of the boxes that represents a packet belonging to different VC includes designation of Ingress and Egress Ports to indicate where the packet is coming from and where it is going to. For example, designation “3.0” means that this packet is arriving at Port #0 (Ingress) and is destined to Port #3 (Egress). Within the Switch, packets are routed and serviced based on Switch internal arbitration mechanisms.

Switch arbitration model defines a required arbitration infrastructure and functionality within a Switch. This functionality is needed to support a set of arbitration policies that control traffic contention for an Egress Port from multiple Ingress Ports.

§ Figure 6-8 shows a conceptual model of a Switch highlighting resources and associated functionality in ingress to egress direction. Note that each Port in the Switch can have the role of an Ingress or Egress Port. Therefore, this figure only shows one particular scenario where the 4-Port Switch in this example has ingress traffic on Port #0 and Port #1, that targets Port #2 as an Egress Port. A different example may show different flow of traffic implying different roles for Ports on the Switch. The PCI Express architecture enables peer-to-peer communication through the Switch and, therefore, possible scenarios using the same example may include multiple separate and simultaneous ingress to egress flows (e.g., Port 0 to Port 2 and Port 1 to Port 3).

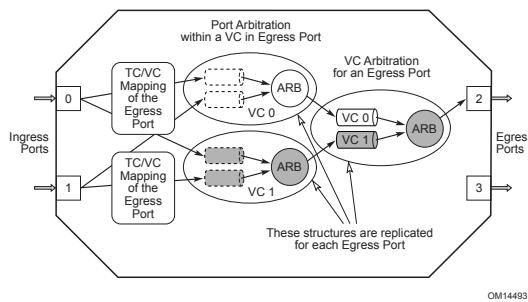


Figure 6-8 Switch Arbitration Structure §

The following two steps conceptually describe routing of traffic received by the Switch on Port 0 and Port 1 and destined to Port 2. First, the target Egress Port is determined based on address/routing information in the TLP header. Secondly,

the target VC of the Egress Port is determined based on the TC/VC map of the Egress Port. Transactions that target the same VC in the Egress Port but are from different Ingress Ports must be arbitrated before they can be forwarded to the corresponding resource in the Egress Port. This arbitration is referred to as the Port Arbitration.

Once the traffic reaches the destination VC resource in the Egress Port, it is subject to arbitration for the shared Link. From the Egress Port point of view this arbitration can be conceptually defined as a simple form of multiplexing where the multiplexing control is based on arbitration policies that are either fixed or configurable/programmable. This stage of arbitration between different VCs at an Egress Port is called the VC Arbitration of the Egress Port.

Independent of VC arbitration policy, a management/control logic associated with each VC must observe transaction ordering and flow control rules before it can make pending traffic visible to the arbitration mechanism.

IMPLEMENTATION NOTE: VC CONTROL LOGIC AT THE EGRESS PORT §

VC control logic at every Egress Port includes:

- VC Flow Control logic
- VC Ordering Control logic

Flow control credits are exchanged between two Ports connected to the same Link. Availability of flow control credits is one of the qualifiers that VC control logic must use to decide when a VC is allowed to compete for the shared Link resource (i.e., Data Link Layer transmit/retry buffer). If a candidate packet cannot be submitted due to the lack of an adequate number of flow control credits, VC control logic must mask the presence of pending packet to prevent blockage of traffic from other VCs. Note that since each VC includes buffering resources for Posted Requests, Non-Posted Requests, and Completion packets, the VC control logic must also take into account availability of flow control credits for the particular candidate packet. In addition, VC control logic must observe ordering rules (see § Section 2.4 for more details) for Posted/Non-Posted/Completion transactions to prevent deadlocks and violation of producer/consumer ordering model.

6.3.3.2 VC Arbitration - Arbitration Between VCs §

This specification defines a default VC prioritization via the VC Identification (VC ID) assignment, i.e., the VC IDs are arranged in ascending order of relative priority in the Virtual Channel Capability structure or Multi-Function Virtual Channel Capability structure. The example in § Figure 6-9 illustrates a Port that supports eight VCs with VC0 treated as the lowest priority and VC7 as the highest priority.

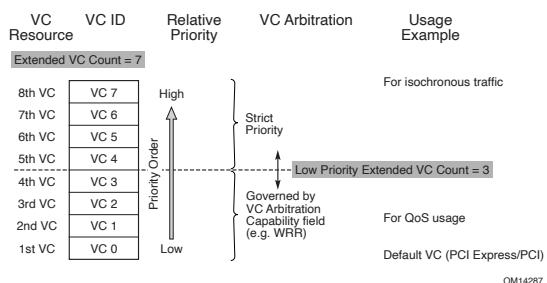


Figure 6-9 VC ID and Priority Order - An Example §

The availability of default prioritization does not restrict the type of algorithms that may be implemented to support VC arbitration - either implementation specific or one of the architecture-defined methods:

- Strict Priority - Based on inherent prioritization, i.e., VC0 = lowest, VC7 = highest
- Round Robin (RR) - Simplest form of arbitration where all VCs have equal priority
- Weighted RR - Programmable weight factor determines the level of service

If strict priority arbitration is supported by the hardware for a subset of the VC resources, software can configure the VCs into two priority groups - a lower and an upper group. The upper group is treated as a strict priority arbitration group while the lower group is arbitrated to only when there are no packets to process in the upper group. § Figure 6-9 illustrates an example configuration that supports eight VCs separated into two groups - the lower group consisting of VC0-VC3 and the upper group consisting of VC4-VC7. The arbitration within the lower group can be configured to one of the supported arbitration methods. The Low Priority Extended VC Count field in the Port VC Capability Register 1 indicates the size of this group. The arbitration methods are listed in the VC Arbitration Capability field in the Port VC Capability Register 2 . Refer to § Section 7.9.1 and § Section 7.9.2 for details. When the Low Priority Extended VC Count field is set to zero, all VCs are governed by the strict-priority VC arbitration; when the field is equal to the Extended VC Count , all VCs are governed by the VC arbitration indicated by the VC Arbitration Capability field.

6.3.3.2.1 Strict Priority Arbitration Model §

Strict priority arbitration enables minimal latency for high-priority transactions. However, there is potential danger of bandwidth starvation should it not be applied correctly. Using strict priority requires all high-priority traffic to be regulated in terms of maximum peak bandwidth and Link usage duration. Regulation must be applied either at the transaction injection ~~↓↓Port/Function~~ ~~↑↑point~~ or within subsequent Egress Ports where data flows contend for a common Link. System software must configure traffic such that lower priority transactions will be serviced at a sufficient rate to avoid transaction timeouts.

Errata: Base 6.3
B823△◀▷

6.3.3.2.2 Round Robin Arbitration Model §

Round Robin arbitration is used to provide, at the transaction level, equal¹³⁸ opportunities to all traffic. Note that this scheme is used where different unordered streams need to be serviced with the same priority.

In the case where differentiation is required, a Weighted Round Robin scheme can be used. The WRR scheme is commonly used in the case where bandwidth regulation is not enforced by the sources of traffic and therefore it is not possible to use the priority scheme without risking starvation of lower priority traffic. The key is that this scheme provides fairness during traffic contention by allowing at least one arbitration win per arbitration loop. Assigned weights regulate both minimum allowed bandwidth and maximum burstiness for each VC during the contention. This means that it bounds the arbitration latency for traffic from different VCs. Note that latencies are also dependent on the maximum packet sizes allowed for traffic that is mapped onto those VCs.

One of the key usage models of the WRR scheme is support for QoS policy where different QoS levels can be provided using different weights.

Although weights can be fixed (by hardware implementation) for certain applications, to provide more generic support for different applications, components that support the WRR scheme are recommended to implement programmable WRR. Programming of WRR is controlled using the software interface defined in Sections 7.9.1 and 7.9.2.

¹³⁸. Note that this does not imply equivalence and fairness in the terms of bandwidth usage.

6.3.3.3 Port Arbitration - Arbitration Within VC §

For Switches, Port Arbitration refers to the arbitration at an Egress Port between traffic coming from other Ingress Ports that is mapped to the same VC. For Root Ports, Port Arbitration refers to the arbitration at a Root Egress Port between peer-to-peer traffic coming from other Root Ingress Ports that is mapped to the same VC. For RCRBs, Port Arbitration refers to the arbitration at the RCRB (e.g., for host memory) between traffic coming from Root Ports that is mapped to the same VC. An inherent prioritization scheme for arbitration among VCs in this context is not applicable since it would imply strict arbitration priority for different Ports. Traffic from different Ports can be arbitrated using the following supported schemes:

- Hardware-fixed arbitration scheme, e.g., Round Robin
- Programmable WRR arbitration scheme
- Programmable Time-based WRR arbitration scheme

Hardware-fixed RR or RR-like scheme is the simplest to implement since it does not require any programmability. It makes all Ports equal priority, which is acceptable for applications where no software-managed differentiation or per-Port-based bandwidth budgeting is required.

Programmable WRR allows flexibility since it can operate as flat RR or if differentiation is required, different weights can be applied to traffic coming from different Ports in the similar manner as described in § [Section 6.3.3.2](#). This scheme is used where different allocation of bandwidth needs to be provided for different Ports.

A Time-based WRR is used for applications where not only different allocation of bandwidth is required but also a tight control of usage of that bandwidth. This scheme allows control of the amount of traffic that can be injected from different Ports within a certain fixed period of time. This is required for certain applications such as isochronous services, where traffic needs to meet a strict deadline requirement. § [Section 6.3.4](#) provides basic rules to support isochronous applications. For more details on time-based arbitration and on the isochronous service as a usage model for this arbitration scheme refer to Appendix A.

6.3.3.4 Multi-Function Devices and Function Arbitration §

The multi-Function arbitration model defines an optional arbitration infrastructure and functionality within a Multi-Function Device. This functionality is needed to support a set of arbitration policies that control traffic contention for the device's Upstream Egress Port from its multiple Functions.

§ [Figure 6-10](#) shows a conceptual model of a Multi-Function Device highlighting resources and associated functionality. Note that each Function optionally contains a VC Extended Capability structure, which if present manages TC/VC mapping, optional Port Arbitration, and optional VC Arbitration, all within the Function. The MFVC Extended Capability structure manages TC/VC mapping, optional Function Arbitration, and optional VC Arbitration for the device's Upstream Egress Port. Together these resources enable enhanced QoS management for Upstream requests. However, unlike a complete Switch with devices on its Downstream Ports, the Multi-Function Device model does not support full QoS management for peer-to-peer requests between Functions or for Downstream requests.

Base 6.4 vs Base 6.3

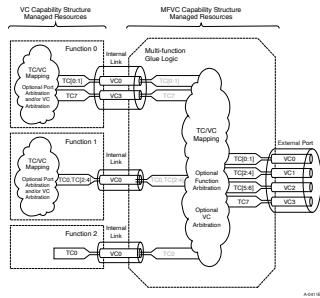


Figure 6-10 Multi-Function Arbitration Model

QoS for an Upstream request originating at a Function is managed as follows. First, a Function-specific mechanism applies a TC to the request. For example, a device driver might configure a Function to tag all its requests with TC7.

Next, if the Function contains a VC Extended Capability structure, it specifies the TC/VC mapping to one of the Function's VC resources (perhaps the Function's single VC resource). In addition, the VC Extended Capability structure supports the enablement and configuration of the Function's VC resources.

If the Function is a Switch and the target VC resource supports Port Arbitration, this mechanism governs how the Switch's multiple Downstream Ingress Ports arbitrate for that VC resource. If the Port Arbitration mechanism supports time-based WRR, this also governs the injection rate of requests from each Downstream Ingress Port.

If the Function supports VC arbitration, this mechanism manages how the Function's multiple VC resources arbitrate for the conceptual internal link to the MFVC resources.

Once a request packet conceptually arrives at MFVC resources, address/routing information in the TLP header determines whether the request goes Upstream or peer-to-peer to another Function. For the case of peer-to-peer, QoS management is left to unarchitected device-specific mechanisms. For the case of Upstream, TC/VC mapping in the MFVC Extended Capability structure determines which VC resource the request will target. The MFVC Extended Capability structure also supports enablement and configuration of the VC resources in the multi-Function glue logic. If the target VC resource supports Function Arbitration, this mechanism governs how the multiple Functions arbitrate for this VC resource. If the Function Arbitration mechanism supports time-based WRR, this governs the injection rate of requests for each Function into this VC resource.

Finally, if the MFVC Extended Capability structure supports VC Arbitration, this mechanism governs how the MFVC's multiple VCs compete for the device's Upstream Egress Port. Independent of VC arbitration policy, management/control logic associated with each VC must observe transaction ordering and flow control rules before it can make pending traffic visible to the arbitration mechanism.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: MULTI-FUNCTION ARBITRATION ERROR BEHAVIOR §

§ Table 6-6 shows the expected error behavior associated with the example topology shown in § Figure 6-10 .

Table 6-6 Multi-Function Arbitration Error Model Example §

| Source | TC | Destination | | | | |
|---------------|-------|-------------|------------|------------|---------------|--|
| | | Function 0 | Function 1 | Function 2 | External Port | |
| Function 0 | 0 | n/a | OK | OK | OK | |
| | 1 | | MF @ F1 | MF @ F2 | OK | |
| | 2 - 6 | | MF @ F0 | MF @ F0 | MF @ F0 | |
| | 7 | | MF @ F1 | MF @ F2 | OK | |
| Function 1 | 0 | OK | n/a | OK | OK | |
| | 1 | MF @ F1 | | MF @ F1 | MF @ F1 | |
| | 2 - 4 | MF @ F0 | | MF @ F2 | OK | |
| | 5 - 7 | MF @ F1 | | MF @ F1 | MF @ F1 | |
| Function 2 | 0 | OK | OK | n/a | OK | |
| | 1 - 7 | MF @ F2 | MF @ F2 | | MF @ F2 | |
| External Port | 0 | OK | OK | OK | n/a | |
| | 1 | OK | MF @ F1 | MF @ F2 | | |
| | 2 - 4 | MF @ F0 | OK | | | |
| | 5 - 6 | MF @ F0 | MF @ F1 | | | |
| | 7 | OK | MF @ F1 | | | |

Legend:

| | |
|---------|--|
| OK | Success |
| MF @ F0 | Malformed TLP, reported at Function 0 |
| MF @ F1 | Malformed TLP, reported at Function 1 |
| MF @ F2 | Malformed TLP, reported at Function 2 |
| n/a | Not Applicable (Function/Port sending to itself) |

IMPLEMENTATION NOTE: MULTI-FUNCTION DEVICES WITHOUT THE MFVC EXTENDED CAPABILITY STRUCTURE §

If a Multi-Function Device lacks an MFVC Extended Capability structure, the arbitration of data flows from different Functions of a Multi-Function Device is beyond the scope of this specification.

However, as stated in this specification, if a Multi-Function Device supports TCs other than TC0 and does not implement an MFVC Extended Capability structure, it is required to implement a single VC Extended Capability structure in Function 0 to provide architected TC/VC mappings for the Link.

6.3.4 Isochronous Support §

Servicing isochronous data transfer requires a system to provide not only guaranteed data bandwidth but also deterministic service latency. The isochronous support mechanisms are defined to ensure that isochronous traffic receives its allocated bandwidth over a relevant period of time while also preventing starvation of the other traffic in the system. Isochronous support mechanisms apply to communication between Endpoint and Root Complex as well as to peer-to-peer communication.

Isochronous service is realized through proper use of mechanisms such as TC transaction labeling, VC data-transfer protocol, and TC-to-VC mapping. End-to-end isochronous service requires software to set up proper configuration along the path between the Requester and the Completer. This section describes the rules for software configuration and the rules hardware components must follow to provide end-to-end isochronous services. More information and background material regarding isochronous applications and isochronous service design guidelines can be found in Appendix A.

6.3.4.1 Rules for Software Configuration §

System software must obey the following rules to configure PCI Express fabric for isochronous traffic:

- Software must designate one or more TCs for isochronous transactions.
- Software must ensure that the Attribute fields of all isochronous requests targeting the same Completer are fixed and identical.
- Software must configure all VC resources used to support isochronous traffic to be serviced (arbitrated) at the requisite bandwidth and latency to meet the application objectives. This may be accomplished using strict priority, WRR, or hardware-fixed arbitration.
- Software should not intermix isochronous traffic with non-isochronous traffic on a given VC.
- Software must observe the Maximum Time Slots capability reported by the Port or RCRB .
- Software must not assign all Link capacity to isochronous traffic. This is required to ensure the requisite forward progress of other non-isochronous transactions to avoid false transaction timeouts.
- Software must limit the Max_Payload_Size for each path that supports isochronous to meet the isochronous latency. For example, all traffic flowing on a path from an isochronous capable device to the Root Complex should be limited to packets that do not exceed the Max_Payload_Size required to meet the isochronous latency requirements.

- Software must set Max_Read_Request_Size of an isochronous-configured device with a value that does not exceed the Max_Payload_Size set for the device.

6.3.4.2 Rules for Requesters §

A Requester requiring isochronous services must obey the following rules:

- The value in the Length field of read requests must never exceed Max_Payload_Size.
- If isochronous traffic targets the Root Complex and the RCRB indicates it cannot meet the isochronous bandwidth and latency requirements without requiring all transactions to set the No Snoop attribute bit, indicated by setting the Reject Snoop Transactions bit, then this bit must be set within the TLP header else the transaction will be rejected.

6.3.4.3 Rules for Completers §

A Completer providing isochronous services must obey the following rules:

- A Completer should not apply flow control induced backpressure to uniformly injected isochronous requests under normal operating conditions.
- A Completer must report its isochronous bandwidth capability in the Maximum Time Slots field in the VC Resource Capability register. Note that a Completer must account for partial writes.
- A Completer must observe the maximum isochronous transaction latency.
- A Root Complex as a Completer must implement at least one RCRB and support time-based Port Arbitration for the associated VCs. Note that time-based Port Arbitration only applies to request transactions.

6.3.4.4 Rules for Switches and Root Complexes §

A Switch providing isochronous services must obey the following rules. The same rules apply to Root Complexes that support isochronous data flows peer-to-peer between Root Ports, abbreviated in this section as “P2P-RC”.

- An isochronous-configured Switch or P2P-RC Port should not apply flow control induced backpressure to uniformly injected isochronous requests under normal operating conditions.
- An isochronous-configured Switch or P2P-RC Port must observe the maximum isochronous transaction latency.
- A Switch or P2P-RC component must support time-based Port Arbitration for each Port that supports one or more VCs capable of supporting isochronous traffic. Note that time-based Port Arbitration applies to request transactions but not to completion transactions.

6.3.4.5 Rules for Multi-Function Devices §

A Multi-Function Device that includes an MFVC Extended Capability structure providing isochronous services must obey the following rules:

- MFVC glue logic configured for isochronous operation should not apply backpressure to uniformly injected isochronous requests from its Functions under normal operating conditions.

- The MFVC Extended Capability structure must support time-based Function Arbitration for each VC capable of supporting isochronous traffic. Note that time-based Function Arbitration applies only to Upstream request transactions; it does not apply to any Downstream or peer-to-peer request transactions, nor to any completion transactions.

A Multi-Function Device that lacks an MFVC Extended Capability structure has no architected mechanism to provide isochronous services for its multiple Functions concurrently.

6.3.5 SVC and VC/MFVC Capability Coexistence §

The Virtual Channel Extended Capability (VC capability) and Multi-Function Virtual Channel Extended Capability (MFVC capability) are referred to as VC/MFVC capabilities. A Port is permitted to implement a Streamlined Virtual Channel Extended Capability (SVC capability) as well as one or more VC/MFVC capabilities. During Link training, VC0 in each VC/MFVC capability will automatically initialize, and VC0 in the SVC capability will remain disabled. This ensures backward compatibility with software that is unaware of SVC.

SVC capability is incompatible with VC/MFVC capabilities, and hardware mechanisms ensure that for a given Port, SVC capability is never enabled at the same time as VC/MFVC capabilities.

SVC-aware software chooses when to use SVC capability instead of VC/MFVC capabilities on a Port that implements both. Before configuring VC/MFVC capabilities, software Clears the Use VC/MFVC bit in the [↓↑SVC Port Status register](#). Doing so immediately clears the VC Enable bit for each VC resource in every VC/MFVC capability (VC Resource Control Register or MFVC VC Resource Control Register). This also Sets the SVC VC Enable bit for VC0 in its SVC Resource Control Register. The Use VC/MFVC bit will remain Clear until the next Conventional Reset. This presents a simplified and consistent operational state, reducing hardware and software complexity.

If a Port implements an SVC capability and no VC/MFVC capabilities, the Use VC/MFVC bit in the [↓↑SVC Port Status register](#) must be hardwired to 0b. During Link training, VC0 in the SVC capability automatically initializes. See the Use VC/MFVC bit description for further required semantics.

6.4 Device Synchronization §

System software requires a “stop” mechanism for ensuring that there are no outstanding transactions for a particular device in a system. For example, without such a mechanism renumbering Bus Numbers during system operation may cause the Requester ID (which includes the Bus Number) for a given device to change while Requests or Completions for that device are still in flight, and may thus be rendered invalid due to the change in the Requester ID. It is also desirable to be able to ensure that there are no outstanding transactions during a Hot-Plug orderly removal.

The details of stop mechanism implementation depend on the device hardware, device driver software, and system software. However, the fundamental requirements which must be supported to allow system software management of the fabric include the abilities to:

- Block the device from generating new Requests
- Block the generation of Requests issued to the device
- Determine that all Requests being serviced by the device have been completed
- Determine that all [↓↑non-posted](#) [↑↑Non-Posted Requests and UIO↑ Requests](#) initiated by the device have completed
- Determine that all posted Requests initiated by the device have reached their destination

Errata: Base 6.3
B834△◀▶

The ability of the driver and/or system software to block new Requests from the device is supported by the Bus Master Enable, SERR# Enable , and Interrupt Disable bits in the Command register (§ Section 7.5.1.1.3) of each device Function, and other such control bits.

Requests issued to the device are generally under the direct control of the driver, so system software can block these Requests by directing the driver to stop generating them (the details of this communication are system software specific). Similarly, Requests serviced by the device are normally under the device driver's control, so determining the completion of such requests is usually trivial.

The ~~↑↓Transactions Pending↓↑↑Transactions Pending↑~~ bit provides a consistent way ~~↑↓on a per Function basis↓~~ for software to determine ~~↑↓that all non posted Requests issued by↓↑↑if↓~~ the ~~↑↓device have been completed↓↑↑Function is expecting one or more Completions↓~~ (see § Section 7.5.3.5).

Errata: Base 6.3
B817△◀▷

Determining that posted Requests have reached their destination is handled by generating a transaction to “flush” any outstanding Requests. Writes to system memory using TC0 will be flushed by host reads of the device, and so require no explicit flush protocol. Writes using TCs other than TC0 require some type of flush synchronization mechanism. The mechanism itself is implementation specific to the device and its driver software. However, in all cases the device hardware and software implementers should thoroughly understand the ordering rules described in § Section 2.4 . This is especially true if the Relaxed Ordering or ID-Based Ordering attributes are set for any Requests initiated by the device.

IMPLEMENTATION NOTE: FLUSH MECHANISMS §

In a simple case such as that of an Endpoint communicating only with host memory through TC0, “flush” can be implemented simply by reading from the Endpoint. If the Endpoint issues writes to main memory using TCs other than TC0, “flush” can be implemented with a memory read on the corresponding TCs directed to main memory. The memory read needs to be performed on all TCs that the Endpoint is using.

If a memory read is used to “flush” outstanding transactions, but no actual read is required, it may be desirable to use the zero-length read semantic described in § Section 2.2.5 .

Peer-to-peer interaction between devices requires an explicit synchronization protocol between the involved devices, even if all communication is through TC0. For a given system, the model for managing peer-to-peer interaction must be established. System software, and device hardware and software must then conform to this model. The requirements for blocking Request generation and determining completion of Requests match the requirements for non-peer interaction, however the determination that Posted Requests have reached peer destination device(s) requires an explicit synchronization mechanism. The mechanism itself is implementation specific to the device, its driver software, and the model used for the establishment and disestablishment of peer communications.

6.5 Locked Transactions §

6.5.1 Introduction §

Locked Transaction support is required to prevent deadlock in systems that use legacy software which causes the accesses to I/O devices. Note that some CPUs may generate locked accesses as a result of executing instructions that implicitly trigger lock. Some legacy software misuses these transactions and generates locked sequences even when exclusive access is not required. Since locked accesses to I/O devices introduce potential deadlocks apart from those mentioned above, as well as serious performance degradation, PCI Express Endpoints are prohibited from supporting

locked accesses, and new software must not use instructions which will cause locked accesses to I/O devices. Legacy Endpoints support locked accesses only for compatibility with existing software.

Only the Root Complex is allowed to initiate Locked Requests on PCI Express. Locked Requests initiated by Endpoints and Bridges are not supported. This is consistent with limitations for locked transaction use outlined in [PCI] (§ Appendix F. - Exclusive Accesses).

This section specifies the rules associated with supporting locked accesses from the Host CPU to Legacy Endpoints, including the propagation of those transactions through Switches and PCI Express/PCI Bridges.

6.5.2 Initiation and Propagation of Locked Transactions - Rules §

Locked transaction sequences are generated by the Host CPU(s) as one or more reads followed by a number of writes to the same location(s). When a lock is established, all other traffic is blocked from using the path between the Root Complex and the locked Legacy Endpoint or Bridge.

- A locked transaction sequence or attempted locked transaction sequence is initiated on PCI Express using the “lock”-type Read Request/Completion (MRdLk/CplDLk) and terminated with the Unlock Message
 - Locked Requests which are completed with a status other than Successful Completion do not establish lock (explained in detail in the following sections)
 - Regardless of the status of any of the Completions associated with a locked sequence, all locked sequences and attempted locked sequences must be terminated by the transmission of an Unlock Message.
 - MRdLk, CplDLk, and Unlock semantics are allowed only for the default Traffic Class (TC0)
 - Only one locked transaction sequence attempt may be in progress at a given time within a single hierarchy domain
- The Unlock Message is sent from the Root Complex down the locked transaction path to the Completer, and may be broadcast from the Root Complex to all Endpoints and Bridges
 - Any device which is not involved in the locked sequence must ignore this Message
- Any violation of the rules for initiation and propagation of locked transactions can result in undefined device and/or system behavior
 - The initiation and propagation of a locked transaction sequence through PCI Express is performed as follows:
- A locked transaction sequence is started with a MRdLk Request
 - Any successive reads for the locked transaction sequence must also use MRdLk Requests
 - The Completions for any MRdLk Request use the CplDLk Completion type for successful Requests, and the CplLk Completion type for unsuccessful Requests
- If any read associated with a locked sequence is completed unsuccessfully, the Requester must assume that the atomicity of the lock is no longer assured, and that the path between the Requester and Completer is no longer locked
- All writes for the locked sequence use MWr Requests
- The Unlock Message is used to indicate the end of a locked sequence
 - A Switch propagates Unlock Messages to the locked Egress Port
- Upon receiving an Unlock Message, a Legacy Endpoint or Bridge must unlock itself if it is in a locked state
 - If not locked, or if the Receiver is a PCI Express Endpoint or Bridge which does not support lock, the Unlock Message is ignored and discarded

6.5.3 Switches and Lock - Rules §

Switches must distinguish transactions associated with locked sequences from other transactions to prevent other transactions from interfering with the lock and potentially causing deadlock. The following rules cover how this is done. Note that locked accesses are limited to TC0, which is always mapped to VC0.

- When a Switch propagates a MRdLk Request from the Ingress Port (closest to the Root Complex) to the Egress Port, it must block all Requests which map to the default Virtual Channel (VC0) from being propagated to the Egress Port
 - If a subsequent MRdLk Request is Received at this Ingress Port addressing a different Egress Port, the behavior of the Switch is undefined

Note: This sort of split-lock access is not supported by PCI Express and software must not cause such a locked access. System deadlock may result from such accesses.
- When the CplDLk for the first MRdLk Request is returned, if the Completion indicates a Successful Completion status, the Switch must block all Requests from all other Ports from being propagated to either of the Ports involved in the locked access, except for Requests which map to non-VC0 on the Egress Port
- The two Ports involved in the locked sequence must remain blocked as described above until the Switch receives the Unlock Message (at the Ingress Port for the initial MRdLk Request)
 - The Unlock Message must be forwarded to the locked Egress Port
 - The Unlock Message may be broadcast to all other Ports
 - The Ingress Port is unblocked once the Unlock Message arrives, and the Egress Port(s) which were blocked are unblocked following the Transmission of the Unlock Message out of the Egress Ports
 - Ports which were not involved in the locked access are unaffected by the Unlock Message

6.5.4 PCI Express/PCI Bridges and Lock - Rules §

The requirements for PCI Express/PCI Bridges are similar to those for Switches, except that, because PCI Express/PCI Bridges use only the default Virtual Channel and Traffic Class, all other traffic is blocked during the locked access. The requirements on the PCI bus side of the PCI Express/PCI Bridge match the requirements for a PCI/PCI Bridge (see [PCI-to-PCI-Bridge] and [PCIe-to-PCI-PCI-X-Bridge]).

6.5.5 Root Complex and Lock - Rules §

A Root Complex is permitted to support locked transactions as a Requester. If locked transactions are supported, a Root Complex must follow the sequence described in § Section 6.5.2 to perform a locked access. The mechanisms used by the Root Complex to interface PCI Express to the Host CPU(s) are outside the scope of this document.

6.5.6 Legacy Endpoints §

Legacy Endpoints are permitted to support locked accesses, although their use is discouraged. If locked accesses are supported, Legacy Endpoints must handle them as follows:

- The Legacy Endpoint becomes locked when it Transmits the first Completion for the first Read Request of the locked access with a Successful Completion status
 - If the completion status is not Successful Completion, the Legacy Endpoint does not become locked

- Once locked, the Legacy Endpoint must remain locked until it receives the Unlock Message
 - While locked, a Legacy Endpoint must not issue any Requests using TCs which map to the default Virtual Channel (VC0)
- Note that this requirement applies to all possible sources of Requests within the Endpoint, in the case where there is more than one possible source of Requests.
- Requests may be issued using TCs which map to VCs other than the default Virtual Channel

6.5.7 PCI Express Endpoints §

PCI Express Endpoints do not support lock. A PCI Express Endpoint must treat a MRdLk Request as an Unsupported Request (see § Chapter 2.).

6.6 PCI Express Reset - Rules §

This section specifies the PCI Express Reset mechanisms. This section covers the relationship between the architectural mechanisms defined in this document and the reset mechanisms defined in this document. Any relationship between the PCI Express Conventional Reset and component or platform reset is component or platform specific (except as explicitly noted).

6.6.1 Conventional Reset §

Conventional Reset includes all reset mechanisms other than Function Level Reset. There are two categories of Conventional Resets: Fundamental Reset and resets that are not Fundamental Reset. This section applies to all types of Conventional Reset.

In all form factors and system hardware configurations, there must, at some level, be a hardware mechanism for setting or returning all Port states to the initial conditions specified in this document - this mechanism is called “Fundamental Reset”. This mechanism can take the form of an auxiliary signal provided by the system to a component or adapter card, in which case the signal must be called PERST#, and must conform to the rules specified in § Section 4.2.5.9.1 . When PERST# is provided to a component or adapter, this signal must be used by the component or adapter as Fundamental Reset. When PERST# is not provided to a component or adapter, Fundamental Reset is generated autonomously by the component or adapter, and the details of how this is done are outside the scope of this document. If a Fundamental Reset is generated autonomously by the component or adapter, and if power is supplied by the platform to the component/adapter, the component/adapter must generate a Fundamental Reset to itself if the supplied power goes outside of the limits specified for the form factor or system.

- There are three distinct types of Conventional Reset: Cold , Warm , and Hot :
 - A Fundamental Reset must occur following the application of power to the component. This is called a Cold Reset .
 - In some cases, it may be possible for the Fundamental Reset mechanism to be triggered by hardware without the removal and re-application of power to the component. This is called a Warm Reset .
 - This document does not specify a means for generating a Warm or Cold Reset .
 - There is an in-band mechanism for propagating Conventional Reset across a Link. This is called a Hot Reset and is described in § Section 4.2.5.9.2 .

There is an in-band mechanism for software to force a Link into Electrical Idle, “disabling” the Link. The Disabled LTSSM state is described in § Section 4.2.6.10 , the Link Disable control bit is described

in § Section 7.5.3.7 , and the Downstream Port Containment mechanism is described in § Section 6.2.11 . Disabling a Link causes Downstream components to undergo a hot reset.

See § Section 2.9 for additional requirements regarding the effects of the Data Link Layer reporting DL_Down status, and how those effects relate to hot reset.

- On exit from any type of Conventional Reset (cold, warm, or hot), all Port registers and state machines must be set to their initialization values as specified in this document, except for sticky registers (see § Section 7.4).
 - Note that, from a device point of view, any type of Conventional Reset (cold, warm, hot, or DL_Down) has the same effect at the Transaction Layer and above as would RST# assertion and deassertion in conventional PCI.
- On exit from a Fundamental Reset, the Physical Layer will attempt to bring up the Link (see § Section 4.2.6). 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 exit from a Conventional Reset, some devices may require additional time before they are able to respond to Requests they receive. Particularly for Configuration Requests it is necessary that components and devices behave in a deterministic way, which the following rules address.

The first set of rules addresses requirements for components and devices:

- A component that supports Link speeds greater than 5.0 GT/s must enter the LTSSM Detect state within 100 ms of the end of Fundamental Reset (Link Training is described in § Section 4.2.5). A component that supports only Link speeds 5.0 GT/s or less must do this within 20 ms. All components are strongly encouraged to minimize this time period. This also applies to Retimers, see § Section 4.3.4.
 - Note: In some systems, it is possible that the components on a Link may exit Fundamental Reset at different times. Each component must observe the requirement to enter the initial active Link Training state within the applicable time period based on the end of Fundamental Reset from its own point of view.
- On the completion of Link Training (entering the DL_Active state, see § Section 3.2), a component must be able to receive and process TLPs and DLLPs.
- Following exit from a Conventional Reset of a device, within 1.0 s the device must be able to receive a Configuration Request and return a Successful Completion if the Request is valid. This period is independent of how quickly Link training completes. If Readiness Notifications mechanisms are used (see § Section 6.22), this period may be shorter, or, with appropriate system support, longer.

The second set of rules addresses requirements placed on the system:

- To allow components to perform internal initialization, system software must wait a specified minimum period following exit from a Conventional Reset of one or more devices before it is permitted to issue Configuration Requests to those devices, unless Readiness Notifications mechanisms are used (see § Section 6.22). System software is also exempted from these minimum waiting period requirements if it knows of the specific device's requirements via means outside the scope of this specification.
- System software is permitted to immediately issue Configuration Requests to a device/Function if Immediate Readiness is indicated by the device/Function (see § Section 3.3 and § Section 7.9.16 for two ways Immediate Readiness support is permitted to be indicated)
- System software is permitted to immediately issue Configuration Requests to a device/Function below a Downstream Port if the DRS Message Received bit is Set.
- Devices that support Flit Mode are required to implement DRS

- Because DRS MUST@FLIT be supported, system software can use the Downstream Component Presence and Flit Mode Status fields in a Downstream Port to determine that DRS is supported by the attached Device. For cases where system software cannot determine that DRS is supported by the attached device, or by the Downstream Port above the attached device:
 - With a Downstream Port that does not support Link speeds greater than 5.0 GT/s, software must wait a minimum of 100 ms following exit from a Conventional Reset before sending a Configuration Request to the device immediately below that Port.
 - With a Downstream Port that supports Link speeds greater than 5.0 GT/s, software must wait a minimum of 100 ms after Link training completes before sending a Configuration Request to the device immediately below that Port. Software can determine when Link training completes by polling the Data Link Layer Link Active bit or by setting up an associated interrupt (see § Section 6.7.3.3). It is strongly recommended for software to use this mechanism whenever the Downstream Port supports it.
 - For a Device that implements the Readiness Time Reporting Extended Capability, and has reported a Reset Time shorter than 100 ms, software is permitted to send a Configuration Request to the Device after waiting the reported Reset Time from Conventional Reset.
 - A system must guarantee that all components intended to be software visible at boot time are ready to receive Configuration Requests within the applicable minimum period based on the end of Conventional Reset at the Root Complex - how this is done is beyond the scope of this specification.
 - It is strongly recommended that software use 100 ms wait periods only if software enables Configuration RRS Software Visibility. Otherwise, Completion timeouts, platform timeouts, or lengthy processor instruction stalls may result. See the Request Retry Status for Configuration Requests Implementation Note in § Section 2.3.1.
 - Unless Readiness Notifications mechanisms are used, the Root Complex and/or system software must allow at least 1.0 s following exit from a Conventional Reset of a device, before determining that the device is broken if it fails to return a Successful Completion status for a valid Configuration Request. This period is independent of how quickly Link training completes.
Note: This delay is analogous to the T_{rhfa} parameter specified for PCI/PCI-X, and is intended to allow an adequate amount of time for devices which require self initialization.
- When attempting a Configuration access to devices on a PCI or PCI-X bus segment behind a PCI Express/PCI(-X) Bridge, the timing parameter T_{rhfa} must be respected.

For this second set of rules, if system software does not have direct visibility into the state of Fundamental Reset (e.g., Hot-Plug; see § Section 6.7), software must base these timing parameters on an event known to occur after the end of Fundamental Reset.

When a Link is in normal operation, the following rules apply:

- If, for whatever reason, a normally operating Link goes down, the Transaction and Data Link Layers will enter the DL_Inactive state (see § Section 2.9 and § Section 3.2.1).
- For any Root or Switch Downstream Port, setting the Secondary Bus Reset bit of the Bridge Control Register associated with the Port must cause a hot reset to be sent (see § Section 4.2.5.9.2).
- For a Switch, the following must cause a hot reset to be sent on all Downstream Ports:
 - Setting the Secondary Bus Reset bit of the Bridge Control Register associated with the Upstream Port
 - The Data Link Layer of the Upstream Port reporting DL_Down status. In Switches that support Link speeds greater than 5.0 GT/s, the Upstream Port must direct the LTSSM of each Downstream Port to the Hot Reset state, but not hold the LTSSMs in that state. This permits each Downstream Port to begin Link training immediately after its hot reset completes. This behavior is recommended for all Switches.

- Receiving a hot reset on the Upstream Port

Certain aspects of Fundamental Reset are specified in this document and others are specific to a platform, form factor and/or implementation. Specific platforms, form factors or application spaces may require the additional specification of the timing and/or sequencing relationships between the components of the system for Fundamental Reset. For example, it might be required that all PCI Express components within a chassis observe the assertion and deassertion of Fundamental Reset at the same time (to within some tolerance). In a multi-chassis environment, it might be necessary to specify that the chassis containing the Root Complex be the last to exit Fundamental Reset.

In all cases where power and PERST# are supplied, the following parameters must be defined:

- T_{pvperl} - PERST# must remain asserted at least this long after power becomes valid
- T_{perst} - When asserted, PERST# must remain asserted at least this long
- T_{fail} - When power becomes invalid, PERST# must be asserted within this time
- $T_{perstslew}$ – The slew rate of PERST# transition to deasserted through its logic input switching range. $T_{perstslew}$ is specified as a minimum of 50 mV/ns unless the form factor specification states otherwise.

Additional parameters may be specified.

In all cases where a reference clock is supplied, the following parameter must be defined:

- $T_{perst-clk}$ - PERST# must remain asserted at least this long after any supplied reference clock is stable

Additional parameters may be specified.

6.6.2 Function Level Reset (FLR) §

The FLR mechanism enables software to quiesce and reset Endpoint hardware with Function-level granularity. Three example usage models illustrate the benefits of this feature:

- In some systems, it is possible that the software entity that controls a Function will cease to operate normally. To prevent data corruption, it is necessary to stop all PCI Express and external I/O (not PCI Express) operations being performed by the Function. Other defined reset operations do not guarantee that external I/O operations will be stopped.
- In a partitioned environment where hardware is migrated from one partition to another, it is necessary to ensure that no residual “knowledge” of the prior partition be retained by hardware, for example, a user’s secret information entrusted to the first partition but not to the second. Further, due to the wide range of Functions, it is necessary that this be done in a Function-independent way.
- When system software is taking down the software stack for a Function and then rebuilding that stack, it is sometimes necessary to return the state to an uninitialized state before rebuilding the Function’s software stack.

Implementation of FLR is optional (not required), but is strongly recommended.

FLR applies on a per Function basis. Only the targeted Function is affected by the FLR operation. The Link state must not be affected by an FLR.

FLR modifies the Function state described by this specification as follows:

- Function registers and Function-specific state machines must be set to their initialization values as specified in this document, except that in the following cases, the values are not affected by FLR:
 - sticky-type registers (ROS , RWS , RW1CS)

Base 6.4 vs Base 6.3

- registers defined as type HwInit
- these other fields or registers:
 - Captured Slot Power Limit Value in the Device Capabilities Register
 - Captured Slot Power Limit Scale in the Device Capabilities Register
 - Max_Payload_Size in the Device Control Register
 - Active State Power Management (ASPM) Control in the Link Control Register
 - Read Completion Boundary (RCB) in the Link Control Register
 - Common Clock Configuration in the Link Control Register
 - Extended Synch in the Link Control Register
 - Enable Clock Power Management in the Link Control Register
 - Hardware Autonomous Width Disable in Link Control Register
 - Hardware Autonomous Speed Disable in the Link Control 2 Register
 - Link Equalization Request 8.0 GT/s in the Link Status 2 Register
 - Enable Lower SKP OS Generation Vector in the Link Control 3 register
 - Lane Equalization Control Register in the Secondary PCI Express Extended Capability structure
 - ↑↑LTR Mechanism Enable bit in the Device Control 2 Register if the LTR Enable Preserved Across FLR bit is Set↑
 - All registers in the Virtual Channel Extended Capability structure
 - All registers in the Multi-Function Virtual Channel Extended Capability structure
 - All registers in the Streamlined Virtual Channel Extended Capability structure
 - All registers in the Data Link Feature Extended Capability structure
 - All registers in the Physical Layer 16.0 GT/s Extended Capability structure
 - All registers in the Physical Layer 32.0 GT/s Extended Capability structure
 - All registers in the Physical Layer 64.0 GT/s Extended Capability structure
 - All registers in the Lane Margining at the Receiver Extended Capability structure
 - All registers in the Flit Logging Extended Capability structure
 - All registers in the Flit Error Injection Extended Capability structure
 - All registers in the Flit Performance Measurement Extended Capability
 - All registers in the NOP Flit Extended Capability
 - CMA-SPDM session state
 - The following registers *MUST*@FLIT also not reset to their initialization values:
 - ARI Control Register in the ARI Extended Capability Structure
 - All registers in the L1 PM Substates Extended Capability structure
 - All registers in the Latency Tolerance Reporting Extended Capability structure
 - All registers in the Precision Time Measurement Extended Capability structure
 - It is strongly recommended that the following registers are also not reset to their initialization values:

 ECN: Base 6.3
 LTR-MFD△◀▷

- All registers in the NPEM Extended Capability structure

Future revisions of this specification may change this recommendation to a requirement.

Note that the controls that enable the Function to initiate requests on PCI Express are cleared, including Bus Master Enable, MSI Enable, and the like, effectively causing the Function to become quiescent on the Link.

Note that Port state machines associated with Link functionality including those in the Physical and Data Link Layers are not reset by FLR, and VC0 remains initialized following an FLR.

- Any outstanding INTx interrupt asserted by the Function must be deasserted by sending the corresponding Deassert_INTx Message prior to starting the FLR.

Note that when the FLR is initiated to a Function of a Multi-Function Device, if another Function continues to assert a matching INTx, no Deassert_INTx Message will be transmitted.

After an FLR has been initiated by writing a 1b to the Initiate Function Level Reset bit, the Function must complete the FLR within 100 ms. If software initiates an FLR when the $\downarrow\downarrow$ Transactions Pending $\downarrow\uparrow\uparrow$ Transactions Pending \uparrow bit is 1b, then software must not initialize the Function until allowing adequate time for any associated Completions to arrive, or to achieve reasonable certainty that any remaining Completions will never arrive. For this purpose, it is recommended that software allow as much time as provided by the pre- FLR value for Completion Timeout on the device. If Completion Timeouts were disabled on the Function when FLR was issued, then the delay is system dependent but must be no less than 100 ms. If Function Readiness Status (FRS - see § Section 6.22.2) is implemented, then system software is permitted to issue Configuration Requests to the Function immediately following receipt of an FRS Message indicating Configuration-Ready, however, this does not necessarily indicate that outstanding Requests initiated by the Function have completed.

Note that upon receipt of an FLR, a device Function may either clear all transaction status including $\downarrow\downarrow$ Transactions Pending $\downarrow\uparrow\uparrow$ Transactions Pending \uparrow or set the Completion Timeout to its default value so that all pending transactions will time out during FLR execution. Regardless, the $\downarrow\downarrow$ Transactions Pending $\downarrow\uparrow\uparrow$ Transactions Pending \uparrow bit must be clear upon completion of the FLR.

Since FLR modifies Function state not described by this specification (in addition to state that is described by this specification), it is necessary to specify the behavior of FLR using a set of criteria that, when applied to the Function, show that the Function has satisfied the requirements of FLR. The following criteria must be applied using Function-specific knowledge to evaluate the Function's behavior in response to an FLR:

- The Function must not give the appearance of an initialized adapter with an active host on any external interfaces controlled by that Function. The steps needed to terminate activity on external interfaces are outside of the scope of this specification.
 - For example, a network adapter must not respond to queries that would require adapter initialization by the host system or interaction with an active host system, but is permitted to perform actions that it is designed to perform without requiring host initialization or interaction. If the network adapter includes multiple Functions that operate on the same external network interface, this rule affects only those aspects associated with the particular Function reset by FLR.
- The Function must not retain within itself software readable state that potentially includes secret information associated with any preceding use of the Function. Main host memory assigned to the Function must not be modified by the Function.
 - For example, a Function with internal memory readable directly or indirectly by host software must clear or randomize that memory.
- The Function must return to a state such that normal configuration of the Function's PCI Express interface will cause it to be $\downarrow\downarrow$ useable $\downarrow\uparrow\uparrow$ usable \uparrow by drivers normally associated with the Function

When an FLR is initiated, the targeted Function must behave as follows:

- The Function must return the Completion for the configuration write that initiated the FLR operation and then initiate the FLR.
- While an FLR is in progress:
 - If a Request arrives, the Request is permitted to be silently discarded (following update of flow control credits) without logging or signaling it as an error.
 - If a Completion arrives, the Completion is permitted to be handled as an Unexpected Completion or to be silently discarded (following update of flow control credits) without logging or signaling it as an error.
 - While a Function is required to complete the FLR operation within the time limit described above, the subsequent Function-specific initialization sequence may require additional time. If additional time is required, the Function must return a Request Retry Status (RRS) Completion Status when a Configuration Request is received after the time limit above. After the Function responds to a Configuration Request with a Completion status other than RRS, it is not permitted to return RRS in response to a Configuration Request until it is reset again.

IMPLEMENTATION NOTE: AVOIDING DATA CORRUPTION FROM STALE COMPLETIONS §

An FLR causes a Function to lose track of any outstanding ~~↓↑non-posted↓↑~~^{↑↑Non-Posted Requests and UIO↑} Requests. Any corresponding Completions that later arrive are referred to as being "stale". If software issues an FLR while there are outstanding Requests, and then re-enables the Function for operation without waiting for potential stale Completions, any stale Completions that arrive afterwards may cause data corruption by being mistaken by the Function as belonging to Requests issued since the FLR.

Errata: Base 6.3
B834△▫

Software can avoid data corruption from stale Completions in a variety of ways. Here's a possible algorithm:

- Software that's performing the FLR synchronizes with other software that might potentially access the Function directly, and ensures such accesses do not occur during this algorithm.
- Software clears the entire Command register, disabling the Function from issuing any new Requests.
- Software polls the ~~↓↑Transactions Pending↓↑~~^{↑↑Transactions Pending↑} bit in the Device Status register either until it is clear or until it has been long enough that software is reasonably certain that Completions associated with any remaining outstanding Transactions will never arrive. On many platforms, the ~~↓↑Transactions Pending↓↑~~^{↑↑Transactions Pending↑} bit will usually clear within a few milliseconds, so software might choose to poll during this initial period using a tight software loop. On rare cases when the ~~↓↑Transactions Pending↓↑~~^{↑↑Transactions Pending↑} bit does not clear by this time, software will need to poll for a much longer platform-specific period (potentially seconds), so software might choose to conduct this polling using a timer-based interrupt polling mechanism.
- Software initiates the FLR.
- Software waits 100 ms.
- Software reconfigures the Function and enables it for normal operation.

6.7 PCI Express Native Hot-Plug §

The PCI Express architecture is designed to natively support both hot-add and hot-removal (“hot-plug”) of cables, add-in cards, and modules. PCI Express native hot-plug provides a “toolbox” of mechanisms that allow different user/operator models to be supported using a self-consistent infrastructure. These mechanisms may be used to implement orderly addition/removal that relies on coordination with the operating system (e.g., traditional PCI hot-plug), as well as async removal, which proceeds without lock-step synchronization with the operating system. This section defines the set of hot-plug mechanisms and specifies how the elements of hot-plug, such as indicators and push buttons, must behave if implemented in a system.

6.7.1 Elements of Hot-Plug §

§ Table 6-7 lists the physical elements comprehended in this specification for support of hot-plug models. A form factor specification must define how these elements are used in that form factor. For a given form factor specification, it is possible that only some of the available hot-plug elements are required, or even that none of these elements are required. In all cases, the form factor specification must define all assumptions and limitations placed on the system or the user by the choice of elements included. Silicon component implementations that are intended to be used only with selected form factors are permitted to support only those elements that are required by the associated form factor(s).

Table 6-7 Elements of Hot-Plug §

| Element | Purpose |
|---|--|
| Indicators | Show the power and attention state of the slot |
| Manually-operated Retention Latch (MRL) | Holds adapter in place |
| MRL Sensor | Allows the Port and system software to detect the MRL being opened |
| Electromechanical Interlock | Prevents removal of adapter from slot |
| Attention Button | Allows user to request hot-plug operations |
| Software User Interface | Allows user to request hot-plug operations |
| Slot Numbering | Provides visual identification of slots |
| Power Controller | Software-controlled electronic component or components that control power to a slot or adapter and monitor that power for fault conditions |
| Out-of-band Presence Detect | Method of determining physical presence of an adapter in a slot that does not rely on the Physical Layer |

6.7.1.1 Indicators §

Two indicators are defined: the Power Indicator and the Attention Indicator. Each indicator is in one of three states: on, off, or blinking. Hot-plug system software has exclusive control of the indicator states by writing the command registers associated with the indicator (with one exception noted below). The indicator requirements must be included in all form factor specifications. For a given form factor, the indicators may be required or optional or not applicable at all.

The hot-plug capable Port controls blink frequency, duty cycle, and phase of the indicators. Blinking indicators must operate at a frequency of between 1 and 2 Hz, with a 50% (+/- 5%) duty cycle. Blinking indicators are not required to be synchronous or in-phase between Ports.

Indicators may be physically located on the chassis or on the adapter (see the associated form factor specification for Indicator location requirements). Regardless of the physical location, logical control of the indicators is by the Downstream Port of the Upstream component on the Link.

The Downstream Port must not change the state of an indicator unless commanded to do so by software, except for platforms capable of detecting stuck-on power faults (relevant only when a power controller is implemented). In the case of a stuck-on power fault, the platform is permitted to override the Downstream Port and force the Power Indicator to be on (as an indication that the adapter should not be removed). The handling by system software of stuck-on faults is optional and not described in this specification. Therefore, the platform vendor must ensure that this feature, if implemented, is addressed via other software, platform documentation, or by other means.

6.7.1.1.1 Attention Indicator §

The Attention Indicator, which must be yellow or amber in color, indicates that an operational problem exists or that the hot-plug slot is being identified so that a human operator can locate it easily.

Table 6-8 Attention Indicator States §

| Indicator Appearance | Meaning |
|----------------------|---|
| Off | Normal - Normal operation |
| On | Attention - Operational problem at this slot |
| Blinking | Locate - Slot is being identified at the user's request |

Attention Indicator Off

The Attention Indicator in the Off state indicates that neither the adapter (if one is present) nor the hot-plug slot requires attention.

Attention Indicator On

The Attention Indicator in the On state indicates that an operational problem exists at the adapter or slot.

An operational problem is a condition that prevents continued operation of an adapter. The operating system or other system software determines whether a specific condition prevents continued operation of an adapter and whether lighting the Attention Indicator is appropriate. Examples of operational problems include problems related to external cabling, adapter, software drivers, and power faults. In general, the Attention Indicator in the On state indicates that an operation was attempted and failed or that an unexpected event occurred.

The Attention Indicator is not used to report problems detected while validating the request for a hot-plug operation. Validation is a term applied to any check that system software performs to assure that the requested operation is viable, permitted, and will not cause problems. Examples of validation failures include denial of permission to perform a hot-plug operation, insufficient power budget, and other conditions that may be detected before a hot-plug request is accepted.

Attention Indicator Blinking

A blinking Attention Indicator indicates that system software is identifying this slot for a human operator to find. This behavior is controlled by a user (for example, from a software user interface or management tool).

6.7.1.1.2 Power Indicator §

The Power Indicator, which must be green in color, indicates the power state of the slot. § Table 6-9 lists the Power Indicator states.

Table 6-9 Power Indicator States §

| Indicator Appearance | Meaning |
|----------------------|--|
| Off | Power Off - Insertion or removal of the adapter is permitted. |
| On | Power On - Insertion or removal of the adapter is not permitted. |
| Blinking | Power Transition - Hot-plug operation is in progress and insertion or removal of the adapter is not permitted. |

Power Indicator Off

The Power Indicator in the Off state indicates that insertion or removal of the adapter is permitted. Main power to the slot is off if required by the form factor. Note that, depending on the form factor, other power/signals may remain on, even when main power is off and the Power Indicator is off. In an example using the [CEM] form factor, if the platform provides Vaux to hot-plug slots and the MRL is closed, any signals switched by the MRL are connected to the slot even when the Power Indicator is off. Signals switched by the MRL are disconnected when the MRL is opened. System software must cause a slot's Power Indicator to be turned off when the slot is not powered and/or it is permissible to insert or remove an adapter. Refer to the appropriate form factor specification for details.

Power Indicator On

The Power Indicator in the On state indicates that the hot-plug operation is complete and that main power to the slot is On and that insertion or removal of the adapter is not permitted.

Power Indicator Blinking

A blinking Power Indicator indicates that the slot is powering up or powering down and that insertion or removal of the adapter is not permitted.

The blinking Power Indicator also provides visual feedback to the operator when the Attention Button is pressed or when hot-plug operation is initiated through the hot-plug software interface.

6.7.1.2 Manually-operated Retention Latch (MRL) §

An MRL is a manually-operated retention mechanism that holds an adapter in the slot and prevents the user from removing the device. The MRL rigidly holds the adapter in the slot so that cables may be attached without the risk of creating intermittent contact. MRLs that hold down two or more adapters simultaneously are permitted in platforms that do not provide MRL Sensors.

6.7.1.3 MRL Sensor §

The MRL Sensor is a switch, optical device, or other type of sensor that reports the position of a slot's MRL to the Downstream Port. The MRL Sensor reports closed when the MRL is fully closed and open at all other times (that is, if the MRL is fully open or in an intermediate position).

If a power controller is implemented for the slot, the slot main power must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open. If signals such as Vaux and SMBus are switched by the MRL, then these signals must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open and must be

restored to the slot when the MRL Sensor indicates that MRL has closed again. Refer to the appropriate form factor specification to identify the signals, if any, switched by the MRL.

Note that the Hot-Plug Controller does not autonomously change the state of either the Power Indicator or the Attention Indicator based on MRL sensor changes.

IMPLEMENTATION NOTE: MRL SENSOR HANDLING §

In the absence of an MRL sensor, for some form factors, out-of-band presence detect may be used to handle the switched signals. In this case, when out-of-band presence detect indicates the absence of an adapter in a slot, the switched signals will be automatically removed from the slot.

If an MRL Sensor is implemented without a corresponding MRL Sensor input on the Hot-Plug Controller, it is recommended that the MRL Sensor be routed to power fault input of the Hot-Plug Controller. This allows an active adapter to be powered off when the MRL is opened.

6.7.1.4 Electromechanical Interlock §

An electromechanical interlock is a mechanism for physically locking the adapter or MRL in place until system software releases it. The state of the electromechanical interlock is set by software and must not change except in response to a subsequent software command. In particular, the state of the electromechanical interlock must be maintained even when power to the hot-plug slot is removed.

The current state of the electromechanical interlock must be reflected at all times in the Electromechanical Interlock Status bit in the Slot Status register, which must be updated within 200 ms of any commanded change. Software must wait at least 1 second after issuing a command to toggle the state of the Electromechanical Interlock before another command to toggle the state can be issued. Systems may optionally expand control of interlocks to provide physical security of the adapter.

6.7.1.5 Attention Button §

The Attention Button is a momentary-contact push button switch, located adjacent to each hot-plug slot or on the adapter that is pressed by the user to initiate a hot-plug operation at that slot. Regardless of the physical location of the button, the signal is processed and indicated to software by hot-plug hardware associated with the Downstream Port corresponding to the slot.

The Attention Button must allow the user to initiate both hot add and hot remove operations regardless of the physical location of the button.

If present, the Power Indicator provides visual feedback to the human operator (if the system software accepts the request initiated by the Attention Button) by blinking. Once the Power Indicator begins blinking, a 5-second abort interval exists during which a second depression of the Attention Button cancels the operation.

If an operation initiated by an Attention Button fails for any reason, it is recommended that system software present an error message explaining the failure via a software user interface or add the error message to a system log.

6.7.1.6 Software User Interface §

System software provides a user interface that allows hot insertions and hot removals to be initiated and that allows occupied slots to be monitored. A detailed discussion of hot-plug user interfaces is operating system specific and is therefore beyond the scope of this document.

On systems with multiple hot-plug slots, the system software must allow the user to initiate operations at each slot independent of the states of all other slots. Therefore, the user is permitted to initiate a hot-plug operation on one slot using either the software user interface or the Attention Button while a hot-plug operation on another slot is in process, regardless of which interface was used to start the first operation.

6.7.1.7 Slot Numbering §

A Physical Slot Identifier (as defined in [PCI-Hot-Plug-1.1], § Section 1.5) consists of an optional chassis number and the physical slot number of the slot. The physical slot number is a chassis unique identifier for a slot. System software determines the physical slot number from registers in the Port. Chassis number 0 is reserved for the main chassis. The chassis number for other chassis must be a non-zero value obtained from a PCI-to-PCI Bridge's Chassis Number register (see [PCI-to-PCI-Bridge], Section 13.4).

Regardless of the form factor associated with each slot, each physical slot number must be unique within a chassis.

IMPLEMENTATION NOTE: DETERMINATION OF SLOT NUMBER INFORMATION §

The Slot Numbering Capability, as defined in the PCI-to-PCI Bridge Specification, is being considered for deprecation in future versions of this specification. System software should use other means to identify physical slot number or chassis ID. Examples include ACPI _DSM for PCI Express Slot Number method (see [Firmware]), ACPI _SUN method (see [UEFI]), and the SMBIOS Type 9 structure (see [SMBIOS]).

6.7.1.8 Power Controller §

The power controller is an element composed of one or more discrete components that acts under control of software to set the power state of the hot-plug slot as appropriate for the specific form factor. The power controller must also monitor the slot for power fault conditions (as defined in the associated form factor specification) that occur on the slot's main power rails and, if supported, auxiliary power rail.

If a power controller is not present, the power state of the hot-plug slot must be set automatically by the hot-plug controller in response to changes in the presence of an adapter in the slot.

The power controller monitors main and auxiliary power faults independently. If a power controller detects a main power fault on the hot-plug slot, it must automatically set its internal main power fault latch and remove main power from the hot-plug slot (without affecting auxiliary power). Similarly, if a power controller detects an auxiliary power fault on the hot-plug slot, it must automatically set its internal auxiliary power fault latch and remove auxiliary power from the hot-plug slot (without affecting main power). Power must remain off to the slot as long as the power fault condition remains latched, regardless of any writes by software to turn on power to the hot-plug slot. The main power fault latch is cleared when software turns off power to the hot-plug slot. The mechanism by which the auxiliary power fault latch is cleared is form factor specific but generally requires auxiliary power to be removed from the hot-plug slot. For example, one form factor may remove auxiliary power when the MRL for the slot is opened while another may require the adapter to be physically removed from the slot. Refer to the associated form factor specifications for specific requirements.

Since the Power Controller Control bit in the Slot Control register reflects the last value written and not the actual state of the power controller, this means there may be an inconsistency between the value of the Power Controller Control bit and the state of the power to the slot in a power fault condition. To determine whether slot is off due to a power fault, software must use the power fault software notification to detect power faults. To determine that a requested power-up operation has otherwise failed, software must use the hot-plug slot power-up time out mechanism described in § Section 6.7.3.3.

Software must not assume that writing to the Slot Control register to change the power state of a hot-plug slot causes an immediate power state transition. After turning power on, software must wait for a Data Link Layer State Changed event, as described in § Section 6.7.3.3. After turning power off, software must wait for at least 1 second before taking any action that relies on power having been removed from the hot-plug slot. For example, software is not permitted to turn off the power indicator (if present) or attempt to turn on the power controller before completing the 1 second wait period.

6.7.2 Registers Grouped by Hot-Plug Element Association §

The registers described in this section are grouped by hot-plug element to convey all registers associated with implementing each element. Register fields associated with each Downstream Port implementing a hot-plug capable slot are located in the Device Capabilities , Slot Capabilities , Slot Control , Slot Status , and Slot Capabilities 2 registers in the PCI Express Capability structure (see § Section 7.5.3). Registers reporting the presence of hot-plug elements associated with the device Function on an adapter are located in the Device Capabilities register (also in the PCI Express Capability structure).

6.7.2.1 Attention Button Registers §

Attention Button Present (Slot Capabilities Register and Device Capabilities Register) - This bit indicates if an Attention Button is electrically controlled by the chassis (Slot Capabilities Register) or by the adapter (Device Capabilities Register).

Attention Button Pressed (Slot Status Register - This bit is Set when an Attention Button electrically controlled by the chassis is pressed.

Attention Button Pressed Enable (Slot Control Register - When Set, this bit enables software notification on an Attention Button Pressed event (see § Section 6.7.3.4).

6.7.2.2 Attention Indicator Registers §

Attention Indicator Present (Slot Capabilities Register and Device Capabilities Register) - This bit indicates if an Attention Indicator is electrically controlled by the chassis (Slot Capabilities Register) or by the adapter (Device Capabilities Register).

Attention Indicator Control (Slot Control Register) - When written, sets an Attention Indicator electrically controlled by the chassis to the written state.

6.7.2.3 Power Indicator Registers §

Power Indicator Present (Slot Capabilities Register and Device Capabilities Register) - This bit indicates if a Power Indicator is electrically controlled by the chassis (Slot Capabilities Register) or by the adapter (Device Capabilities Register).

Power Indicator Control (Slot Control Register) - When written, sets a Power Indicator electrically controlled by the chassis to the written state.

6.7.2.4 Power Controller Registers §

Power Controller Present (Slot Capabilities Register) - This bit indicates if a Power Controller is implemented.

Power Controller Control (Slot Control Register) - Turns the Power Controller on or off according to the value written.

Power Fault Detected (Slot Status Register) - This bit is Set when a power fault is detected at the slot or the adapter.

Power Fault Detected Enable (Slot Control Register) - When Set, this bit enables software notification on a power fault event (see § Section 6.7.3.4).

6.7.2.5 Presence Detect Registers §

In-Band PD Disable Supported (Slot Capabilities 2 Register) - This bit indicates if the slot supports the disabling of in-band presence detect, which allows the out-of-band presence detect state to be reported independently of the in-band presence detect state.

In-Band PD Disable (Slot Control Register) - When Set, this bit disables the in-band presence detect mechanism from affecting the Presence Detect State bit, allowing that bit to be dedicated to reporting out-of-band presence detect.

Presence Detect State (Slot Status Register) - This bit indicates the presence of an adapter in the slot.

Presence Detect Changed (Slot Status Register) - This bit is Set when a presence detect state change is detected.

Presence Detect Changed Enable (Slot Control Register) - When Set, this bit enables software notification on a presence detect changed event (see § Section 6.7.3.4).

6.7.2.6 MRL Sensor Registers §

MRL Sensor Present (Slot Capabilities Register) - This bit indicates if an MRL Sensor is implemented.

MRL Sensor Changed (Slot Status Register) - This bit is Set when the value of the MRL Sensor state changes.

MRL Sensor Changed Enable (Slot Control Register) - When Set, this bit enables software notification on a MRL Sensor changed event (see § Section 6.7.3.4).

MRL Sensor State (Slot Status Register) - This register reports the status of the MRL Sensor if one is implemented.

6.7.2.7 Electromechanical Interlock Registers §

Electromechanical Interlock Present (Slot Capabilities Register) - This bit indicates if an Electromechanical Interlock is implemented.

Electromechanical Interlock Status (Slot Status Register) - This bit reflects the current state of the Electromechanical Interlock.

Electromechanical Interlock Control (Slot Control Register) - This bit when set to 1b toggles the state of the Electromechanical Interlock.

6.7.2.8 Command Completed Registers §

No Command Completed Support (Slot Capabilities Register) - This bit when set to 1b indicates that this slot does not generate software notification when an issued command is completed by the Hot-Plug Controller.

Command Completed (Slot Status Register) - This bit is Set when the Hot-Plug Controller completes an issued command and is ready to accept the next command.

Command Completed Interrupt Enable (Slot Control Register) - When Set, this bit enables software notification (see § Section 6.7.3.4) when a command is completed by the hot-plug control logic.

6.7.2.9 Port Capabilities and Slot Information Registers §

Slot Implemented (PCI Express Capabilities Register) - When Set, this bit indicates that the Link associated with this Downstream Port is connected to a slot.

Physical Slot Number (Slot Capabilities Register) - This hardware initialized field indicates the physical slot number attached to the Port.

Hot-Plug Capable (Slot Capabilities Register) - When Set, this bit indicates this slot is capable of supporting hot-plug.

Hot-Plug Surprise (Slot Capabilities Register) - When Set, this bit indicates that the Hot-Plug Surprise mechanism for handling async removal is enabled for this slot. See § Section 6.7.6 .

6.7.2.10 Hot-Plug Interrupt Control Register §

Hot-Plug Interrupt Enable (Slot Control Register) - When Set, this bit enables generation of the hot-plug interrupt on enabled hot-plug events.

6.7.3 PCI Express Hot-Plug Events §

A Downstream Port with hot-plug capabilities supports the following hot-plug events:

- Slot Events:
 - Attention Button Pressed
 - Power Fault Detected
 - MRL Sensor Changed
 - Presence Detect Changed
- Command Completed Events
- Data Link Layer State Changed Events

Each of these events has a status field, which indicates that an event has occurred but has not yet been processed by software, and an enable field, which indicates whether the event is enabled for software notification. Some events also have a capability field, which indicates whether the event type is supported on the Port. The grouping of these fields by event type is listed in § Section 6.7.2 , and each individual field is described in § Section 7.5.3 .

6.7.3.1 Slot Events §

A Downstream Port with hot-plug capabilities monitors the slot it controls for the slot events listed above. When one of these slot events is detected, the Port indicates that the event has occurred by setting the status field associated with the event. At that point, the event is pending until software clears the status field.

Once a slot event is pending on a particular slot, all subsequent events of that type are ignored on that slot until the event is cleared. The Port must continue to monitor the slot for all other slot event types and report them as they occur.

If enabled through the associated enable field, slot events must generate a software notification. If the event is not supported on the Port as indicated by the associated capability field, software must not enable software notification for the event. The mechanism by which this notification is reported to software is described in § Section 6.7.3.4 .

6.7.3.2 Command Completed Events §

Since changing the state of some hot-plug elements may not happen instantaneously, PCI Express supports hot-plug commands and command completed events. All hot-plug capable Ports are required to support hot-plug commands and, if the capability is reported, command completed events.

Software issues a command to a hot-plug capable Downstream Port by issuing a write transaction that targets any portion of the Port's Slot Control register. A single write to the Slot Control register is considered to be a single command, even if the write affects more than one field in the Slot Control register. In response to this transaction, the Port must carry out the requested actions and then set the associated status field for the command completed event. The Port must process the command normally even if the status field is already set when the command is issued. If a single command results in more than one action being initiated, the order in which the actions are executed is unspecified. All actions associated with a single command execution must not take longer than 1 second.

If command completed events are not supported as indicated by a value of 1b in the No Command Completed Support field of the Slot Capabilities register, a hot-plug capable Port must process a write transaction that targets any portion of the Port's Slot Control register without any dependency on previous Slot Control writes. Software is permitted to issue multiple Slot Control writes in sequence without any delay between the writes.

If command completed events are supported, then software must wait for a command to complete before issuing the next command. However, if the status field is not set after the 1 second limit on command execution, software is permitted to repeat the command or to issue the next command. If software issues a write before the Port has completed processing of the previous command and before the 1 second time limit has expired, the Port is permitted to either accept or discard the write. Such a write is considered a programming error, and could result in a discrepancy between the Slot Control register and the hot plug element state. To recover from such a programming error and return the controller to a consistent state, software must issue a write to the Slot Control register which conforms to the command completion rules.

If enabled through the associated enable field, the completion of a command must generate a software notification. The exception to this rule is a command that occurs as a result of a write to the Slot Control register that disables software notification of command completed events. Such a command must be processed as described above, but must not generate a software notification.

6.7.3.3 Data Link Layer State Changed Events §

The Data Link Layer State Changed event provides an indication that the state of the Data Link Layer Link Active bit in the Link Status Register has changed. Support for Data Link Layer State Changed events and software notification of these events are required for hot-plug capable Downstream Ports. If this event is supported, the Port sets the status field associated with the event when the value in the Data Link Layer Link Active bit changes.

This event allows software to indirectly determine when power has been applied to a newly hot-plugged adapter. Software must wait for 100 ms after the Data Link Layer Link Active bit reads 1b before initiating a configuration access to the hot added device (see § Section 6.6). Software must allow 1 second after the Data Link Layer Link Active bit reads 1b before it is permitted to determine that a hot plugged device which fails to return a Successful Completion for a Valid Configuration Request is a broken device (see § Section 6.6).

The Data Link Layer State Changed event must occur within 1 second of the event that initiates the hot-insertion. If a power controller is supported, the time out interval is measured from when software initiated a write to the Slot Control register to turn on the power. If the Power Disable mechanism is supported, the time out interval is measured from when that mechanism is deasserted (power is restored). If neither mechanism is supported, the time out interval is measured from presence detect slot event. Software is allowed to time out on a hot add operation if the Data Link Layer State Changed event does not occur within 1 second. The action taken by software after such a timeout is implementation specific.

6.7.3.4 Software Notification of Hot-Plug Events §

A hot-plug capable Downstream Port must support generation of an interrupt on a hot-plug event. As described in §§ Sections 6.7.3.1 and §§ Section 6.7.3.1 and §§ Sections 6.7.3.2 and §§ Section 6.7.3.2 each hot-plug event has both an enable bit for interrupt generation and a status bit that indicates when an event has occurred but has not yet been processed by software. There is also a Hot-Plug Interrupt Enable bit in the Slot Control register that serves as a master enable/disable bit for all hot-plug events.

If the Port is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as the following conditions are satisfied:

- The Interrupt Disable bit in the Command register is set to 0b.
- The Hot-Plug Interrupt Enable bit in the Slot Control register is set to 1b.
- At least one hot-plug event status bit in the Slot Status register and its associated enable bit in the Slot Control register are both set to 1b.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If the Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- The Hot-Plug Interrupt Enable bit in the Slot Control register is set to 1b.
- At least one hot-plug event status bit in the Slot Status register and its associated enable bit in the Slot Control register are both set to 1b.

Note that PME and Hot-Plug Event interrupts (when both are implemented) always share the same MSI or MSI-X vector, as indicated by the Interrupt Message Number field in the PCI Express Capabilities Register.

The Port may optionally send an MSI when there are hot-plug events that occur while interrupt generation is disabled, and interrupt generation is subsequently enabled.

If wake generation is required by the associated form factor specification, a hot-plug capable Downstream Port must support generation of a wakeup event (using the PME mechanism) on hot-plug events that occur when the system is in a sleep state or the Port is in device state D1, D2, or D3_{Hot}.

Software enables a hot-plug event to generate a wakeup event by enabling software notification of the event as described in § Section 6.7.3.1. Note that in order for software to disable interrupt generation while keeping wakeup

generation enabled, the Hot-Plug Interrupt Enable bit must be cleared. For form factors that support wake generation, a wakeup event must be generated if all three of the following conditions occur:

- The status register for an enabled event transitions from Clear to Set
- The Port is in device state D1, D2, or D3_{Hot}, and
- The PME_En bit in the Port's Power Management Control/Status register is Set

Note that the Hot-Plug Controller generates the wakeup on behalf of the hot-plugged device, and it is not necessary for that device to have auxiliary (or main) power.

6.7.4 System Firmware Intermediary (SFI) Support §

The System Firmware Intermediary (SFI) Capability is an optional normative feature of a Downstream Port. Some SFI functionality is focused on hot-pluggable slots, as indicated by the Hot-Plug Capable bit in the Slot Capabilities register being Set, while some SFI functionality is useful outside that context. If a Downstream Port supports an SFI Extended Capability structure, the following bits must be Set:

- Data Link Layer Link Active Reporting Capable bit in the Link Capabilities register
- DRS Supported bit in the Link Capabilities 2 register
- ERR_COR Subclass Capable bit in the Device Capabilities register

As originally introduced, SFI is focused on system firmware running on the host CPU(s) as the intermediary. Enhanced SFI (eSFI) adds important new functionality for improved support of the BMC as the intermediary, as well as increased protection against malicious components or adapters. Enhanced SFI also makes SFI CAM support optional instead of mandatory for Root Ports, easing the implementation of eSFI on Root Ports intended for platforms that have no need to support system firmware running on the host CPU(s) as the intermediary.

ECN: Base 6.3
eSFI△↔

6.7.4.1 SFI ERR_COR Event Signaling §

The SFI Extended Capability has no support for generating INTx or MSI/MSI-X interrupts, since the capability is intended for use by system firmware.

A Downstream Port with SFI must support ERR_COR signaling, regardless of whether it supports Advanced Error Reporting (AER) or not. SFI ERR_COR event signaling is enabled independently by the SFI OOB PD Changed Enable, SFI DLL State Changed Enable, and SFI DRS Signaling Enable bits in the SFI Control Register. These events are indicated by the SFI OOB PD Changed, SFI DLL State Changed, and SFI DRS Received bits in the SFI Status Register.

If the Correctable Error Reporting Enable bit in the Device Control Register is Set, the Port must send an ERR_COR Message each time one of the enabled conditions becomes satisfied. SFI ERR_COR event signaling must not Set the Correctable Error Detected bit in the Device Status Register, since this event is not handled as an error.

IMPLEMENTATION NOTE: ERR_COR SIGNALING FOR DPC DL_ACTIVE VS. SFI DLL STATE CHANGED §

DPC implements ERR_COR signaling for DL_Active , whereas SFI implements ERR_COR signaling for SFI DLL State Changed , which are related but non-identical conditions. The DL_Active condition occurs when the Data Link Layer Link Active bit in the Link Status register changes from 0b to 1b, and this bit can be masked by the SFI DLL State Mask bit in the SFI Control register . The SFI DLL State Changed condition occurs when the SFI DLL State bit in the SFI Status Register changes its value either by becoming Set or becoming Clear, and this condition is always based on the actual Data Link Layer state.

6.7.4.2 SFI Downstream Port Filtering (DPF) §

Downstream Port Filtering (DPF) is a mechanism where a Downstream Port can handle filter, or block, specified Request TLPs that target Components below it as if passing through the Link Downstream Port. A filtered Request TLP is in DL_Down . See handled as an Unsupported Request and a filtered Completion TLP is silently discarded (following update of flow control credits for upstream flowing TLPs).¹

ECN: Base 6.3
eSFI△↔

DPF has two supports up to three modes of filtering. The first two modes filter Downstream Request TLPs that target Components below are passing through the Downstream Port. Port other than Configuration Request TLPs generated by that Port's SFI CAM mechanism.¹ The first mode filters all such Request TLPs; the second mode filters only Downstream Configuration Request TLPs not generated by the Port's SFI CAM mechanism.¹ Other TLPs must not be filtered or blocked by DPF.¹ These two modes.¹

ECN: Base 6.3
eSFI△↔

Enhanced SFI also supports a third mode of filtering that allows Upstream and Downstream MCTP VDMs to be passed through the Downstream Port. In addition, this third mode does not filter Configuration Request TLPs generated by PCIe-MI accesses to the Port's SFI CAM mechanism or the Completion TLPs for those Configuration Request TLPs. All other TLPs are filtered.¹

ECN: Base 6.3
eSFI△↔

One key use case for DPF is guaranteeing that asynchronous system software activities like bus scans do not unintentionally send Configuration Requests to devices that are not yet ready following a Conventional Reset, since such accesses result in undefined hardware behavior. See § Section 6.6.1 .

Another key use case for DPF is supporting firmware first functionality, enabling system firmware, firmware running on the host CPU(s), when notified of an async hot add, to configure the newly added device before making the device visible to the operating system. For this use case, the SFI CAM mechanism enables the Downstream Port itself to generate Configuration Request TLPs targeting Downstream Components, and those TLPs and their corresponding Completion TLPs are not filtered or blocked by the DPF mechanism. See § Section 6.7.4.3 , § Section 7.9.20.5 , and § Section 7.9.20.6 .

ECN: Base 6.3
eSFI△↔

The use case for the third DPF mode (block all TLPs except MCTP VDMs and PCIe-MI initiated SFI CAM accesses) is for enabling BMC management of components or adapters using [MCTP-VDM] and the SFI CAM without permitting other PCIe traffic, which malicious components or adapters might use as attack vectors against the system. Key use case examples include authenticating a component or adapter with Component Measurement and Authentication (CMA-SPDM) and configuring a component or adapter with PCIe-MI

ECN: Base 6.3
eSFI△↔

6.7.4.3 SFI CAM §

The SFI Configuration Access Method (CAM) provides a means for SFI-aware system firmware to have the Downstream Port proxy (pass through) Configuration Requests targeting Components below the Downstream Port when DPF is enabled. ~~If supported, the SFI CAM is always enabled.~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

~~The following rules apply to SFI CAM support (see the SFI CAM Unsupported bit and IMPLEMENTATION NOTE: SFI CAM SUPPORT CONSIDERATIONS).~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

- ~~For Root Ports that support eSFI, SFI CAM support is optional when accessed using Configuration Read Requests or Configuration Writes Requests.~~
- ~~For Root Ports that do not support eSFI, SFI CAM support is mandatory when accessed using Configuration Read Requests or Configuration Writes Requests.~~
- ~~For Switch Downstream Ports that support SFI or eSFI, SFI CAM support is mandatory when accessed using Configuration Read Requests or Configuration Writes Requests.~~
- ~~For all Downstream Ports that support SFI or eSFI, SFI CAM support is optional when accessed using PCIe-MI.~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

~~If the SFI RO In-Band bit is Set, Configuration Read Requests and Configuration Write Requests to the SFI CAM are impacted. See § Section 6.7.4.8 for more detail.~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

To use the SFI CAM ~~in cases where the SFI CAM is supported and writeable~~, software first writes to the SFI CAM Address Register, specifying the target Configuration address. Software then reads or writes the SFI CAM Data Register to cause a proxied Configuration Request to be generated and transmitted to the Downstream Component.

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

The following rules ~~apply when the SFI CAM is accessed using Configuration Request TLPs:~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

- All TLP fields used for the proxied Configuration Request are identical to those in the Configuration Request that targeted the SFI CAM Data Register, with the following exceptions:
 - The target Bus Number, Device Number, and Function Number come from the SFI CAM Address Register.
 - The Extended Register Number and Register Number come from the SFI CAM Address Register.
 - The LCRC is regenerated.
 - If present, the ECRC is regenerated.
- ~~If a Completion TLP flowing Upstream is permitted to pass through the Downstream Port (e.g., the Completion is not blocked due to the SFI Downstream Port Filtering field being set to a value of 11b), the Completion TLP must be passed through the Downstream Port unmodified.~~
- ~~If there is a detected error associated with the proxied Configuration Request, this is a reported error associated with the Downstream Port implementing the SFI CAM (see § Section 6.2).~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

~~The following rules apply when the SFI CAM is accessed using PCIe-MI:~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

- ~~All TLP fields used for the proxied Configuration Request are populated by the Port as necessary to satisfy the corresponding PCIe-MI access.~~

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

- ↑↑↑If there is a detected error associated with the proxied Configuration Request, this is a reported error in the PCIe-MI Completion ↑

ECN: Base 6.3
eSFI△↔

↑↑↑The following rules apply when the SFI CAM is accessed using Configuration Request TLPs or PCIe-MI :↑

- The SFI CAM must not apply the Completion Timeout mechanism to the Request.
- ↓↓↓System firmware↓↑↑The intermediary↓ must ensure that between the time it writes to the SFI CAM Address Register and its subsequent read or write of the SFI CAM Data Register completes, no other threads modify the SFI CAM Address Register ; otherwise, the result is undefined.

ECN: Base 6.3
eSFI△↔

↓↓↓If there is↓

IMPLEMENTATION NOTE: USE OF ASSIGNED BUS NUMBERS WITH THE SFI CAM §

↑↑↑When↑ a ↑↑Downstream Port has DPF enabled, the SFI CAM , if supported, can be used by SFI-aware system firmware to configure and access the sub-hierarchy below the Port without other software being able to do so. While the Bus Number configuration below the Port is generally not visible to other software, Bus Numbers configured for use below the Port should be limited to those already assigned to the Port since TLPs coming Upstream through the Port may contain IDs with the configured Bus Numbers. If any errors are↑ detected ↓↓↓error associated↓ ↑↑and logged↑ with ↑↑↑those TLPs,↑ the ↑↑proxied Configuration Request, this is a reported error associated↓ ↑↑↑Bus Numbers can become visible to other software, creating confusion if they overlap↑ with ↑↑↑Bus Numbers used elsewhere in↑ the ↑↑↑system.↑

ECN: Base 6.3
eSFI△↔

IMPLEMENTATION NOTE: SFI CAM SUPPORT CONSIDERATIONS §

↑↑↑SFI as originally architected prior to eSFI support being architected requires implementing the SFI CAM and was intended for use by system firmware running on the host CPU(s) to configure Functions hidden from the view of the operating system running on the host CPU(s) using SFI↑ Downstream Port ↑↑implementing↑ ↑↑Filtering. If eSFI is supported, SFI CAM support is optional under certain conditions. See the SFI CAM Unsupported bit. By permitting↑ the SFI CAM ↑↑(see↑ ↑↑to be optional, this eases the implementation of↑ ↑↑eSFI . However, there are other considerations besides this key benefit.↑

ECN: Base 6.3
eSFI△◀▷

↑↑Enterprise and cloud systems use BMCs extensively and may require support for a BMC to access the SFI CAM (e.g., using PCIe-MI) to generate Configuration Read Requests and Configuration Write Requests to Functions below the Port, allowing a BMC to configure such Functions, which may be necessary if the Functions below the Port do not support a way for the BMC to access the Functions' Configuration Space directly (e.g., using PCIe-MI). ↑↑Completions flowing Upstream must

↑Other systems, e.g., those used for client, mobile, embedded, or automotive, may find it more suitable or even necessary to use the operating system running on the host CPU(s) as the intermediary. This specification does not architect such use of SFI, but such use is not forbidden. In fact, the PCI Firmware Specification architects an OSC mechanism for system firmware/operating system negotiation of SFI ownership, though it recommends that general-purpose operating systems never request ownership, and general-purpose platforms never grant ownership to an operating system if requested.↑

↑ Additionally, any system software running on the host CPU(s) developed for SFI as originally architected may assume that the SFI CAM is implemented and supported, and such software may break if an unimplemented SFI CAM might be passed through mistaken as being fully functional. If an eSFI-capable Port implementation does not support the SFI CAM, the SFI CAM registers are required to behave “as if” the Link below the Downstream Port unmodified. is in DL_Down. See the SFI CAM Data register. This mandatory semantic helps reduce the risk of non-eSFI-aware software breaking when the SFI CAM isn’t supported.

↑Since system firmware running on the host CPU(s) is usually customized for the system's Root Complex(es), there is less concern for non-eSFI-aware system firmware running on the host CPU(s) possibly needing to be modified to accommodate eSFI-capable Root Ports that do not support the SFI CAM .↑

^{↑↑}See the SFI Hidden In-Band bit and the SFI RO In-Band bit for other approaches to avoid implementation complexities due to SFI CAM .[↑]

IMPLEMENTATION NOTE: SERIALIZED USE OF THE SFI CAM ADDRESS AND DATA REGISTERS §

As described above, system firmware must ensure that between the time it writes to the SFI CAM Address Register and its subsequent read or write of the SFI CAM Data Register completes, no other threads modify the SFI CAM Address Register. For example, a semaphore or other synchronization mechanism can be used to ensure this serialization.

For platforms where a processor store instruction to Configuration Space is effectively posted, software must still ensure that the resulting Configuration Write completes before another software thread modifies the SFI CAM Data Register. On such platforms, the mechanism for determining when a Configuration Write completes is platform specific.

Given appropriate serialization, the SFI CAM works correctly with Configuration Requests that result in RRS Completions, even when the Root Complex automatically re-issues the Configuration Request as a new Request. The re-issued Configuration Request will again be sent to the SFI CAM Data Register, and the associated Downstream Port will again generate a Configuration Request targeting the Downstream Component. As long as the SFI CAM Address Register isn't modified by other software until the Configuration Request completes, the sequence can repeat indefinitely until a non-RRS Completion is returned or a Completion Timeout occurs.

When Configuration RRS Software Visibility is enabled, the SFI CAM still works correctly with Configuration Requests that result in RRS Completions. Any Completions with a RRS Completion Status flow back to the original Requester, which handles them as required by Configuration RRS Software Visibility semantics. See § Section 2.3.2.

↑↓ When a Downstream Port has DPF enabled, the SFI CAM can be used by SFI aware system firmware to configure and access the sub hierarchy below the Port without other software being able to do so. While the Bus Number configuration below the Port is generally not visible to other software, Bus Numbers configured for use below the Port should be limited to those already assigned to the Port since TLPs coming Upstream through the Port may contain IDs with the configured Bus Numbers. If any errors are detected and logged with those TLPs, the Bus Numbers can become visible to other software, creating confusion if they overlap with Bus Numbers used elsewhere in the system. ↓

6.7.4.4 SFI Interactions with Readiness Notifications §

The ↑↓SFI Extended Capability↑↑SFI Extended Capability↑ is able to mask the reporting of received Device Readiness Status (DRS) Messages as well as emulate them being received. This functionality is useful when SFI's Downstream Port Filtering (DPF) mechanism is being used to block operating system visibility of a device or sub-hierarchy below the Downstream Port.

Rules:

- When the SFI DRS Mask bit is Set, the DRS Message Received bit in the Link Status 2 Register value must be 0b.
- The SFI DRS Received bit must always indicate the actual state of the DRS Message Received condition.
- When the SFI DRS Mask bit is Clear and a 1b is written to the SFI DRS Trigger bit, the Downstream Port must behave as if a DRS Message was received.

IMPLEMENTATION NOTE: SFI TRANSPARENT OPTIMIZATIONS FOR DEVICE READINESS §

Certain devices may need more time to become Configuration-Ready following a hot-add operation than permitted. See § Section 6.6.1.

If system firmware is aware of such devices, it can use the SFI DPF mechanism to block operating system visibility of a newly added device, wait the necessary amount of time for the device to become Configuration-Ready, and then expose the device to the operating system.

To avoid the operating system from unnecessarily waiting additional time for the newly exposed device to become Configuration-Ready, system firmware can use the SFI DRS Trigger bit to have the Downstream Port emulate the reception of a DRS Message. An operating system that supports DRS can then immediately discover and configure the newly exposed device.

The newly exposed device doesn't necessarily need to be DRS capable itself. Since an Upstream Port is expressly permitted to send DRS Messages even when its DRS Supported bit is Clear, the Downstream Port above it can legitimately emulate receiving a DRS Message from it even if it is incapable of sending DRS Messages.

It should also be noted that in cases where system firmware is aware of a device becoming Configuration-Ready early, system firmware can expose this to the operating system using the SFI DRS Trigger mechanism.

Although SFI is not intended to be used by operating system software, it is recommended that operating systems used in platforms supporting SFI implement support for DRS, so that the system as a whole can have the benefits of this optimized Device Readiness timing.

IMPLEMENTATION NOTE: SFI DPF AND FUNCTION READINESS STATUS (FRS) MESSAGES §

Downstream Port Filtering (DPF) does not affect the generation or propagation of FRS Messages. No FRS Messages are generated by a device when it becomes ready as part of an async hot-add operation. However, if system firmware performs operations on a device that result in FRS events, the resulting FRS Messages may be visible to the operating system. See § Section 2.2.8.6.3 and § Section 6.22.2.

6.7.4.5 SFI Suppression of Hot-Plug Surprise Functionality §

If a slot supports Hot-Plug Surprise (HPS) functionality as indicated by the Hot-Plug Surprise bit in the Slot Capabilities Register being Set, the SFI HPS Suppress bit in the SFI Control Register can be used to force the Hot-Plug Surprise bit to be Clear, and disable the associated Hot-Plug Surprise functionality.

HPS suppression is useful when a Downstream Port / slot combination supports both HPS and Downstream Port Containment (DPC). DPC is not recommended for concurrent use with HPS, so if a slot has HPS capability enabled, DPC should not be enabled. If software wishes to use DPC, software should first Set the SFI HPS Suppress bit in order to disable HPS functionality, allowing DPC to function properly.

IMPLEMENTATION NOTE: SOFTWARE NEGOTIATION OF HOT-PLUG SURPRISE FUNCTIONALITY §

Assuming that system firmware owns the [↑↓SFI Extended Capability↓↑↑SFI Extended Capability↑](#) structure, it is recommended that for backward compatibility with older operating systems, Hot-Plug Surprise functionality be enabled by default on slots supporting async removal. Then, if the slot also supports DPC and the operating system wishes to use it instead, the operating system will request that HPS be suppressed by system firmware, and system firmware will determine whether to Set or Clear the [SFI HPS Suppress](#) bit.

6.7.4.6 ↑↓SFI Quarantine Mode§

ECN: Base 6.3
eSFI△◀▷

[↑↓The SFI Quarantine Mode field and the SFI Quarantine bit provide an intermediary \(e.g., a BMC\) with the ability to configure an eSFI-capable Port to be quarantined \(i.e., Functions below the Port are hidden from system software running on the host CPU\(s\) and blocked from injecting traffic into the platform over the Link\) under a variety of conditions such as Conventional Reset of the Port, hot removal of an adapter connected to the Port, or the LTSSM of the Link below the Port reporting DL_Down status. Quarantining of a Port is accomplished by directing the LTSSM of the Link below the Port to the Disabled state and for hot-plug use cases, masking the presence detect state by keeping the Presence Detect State bit in the Slot Status Register Cleared. Once quarantined, a Port remains quarantined until the SFI quarantine mode is disabled \(i.e., the SFI Quarantine bit is Cleared\).↑](#)

[↑↓A Port transitions to SFI quarantine when the SFI Quarantine bit is Set which occurs in the following conditions.↑](#)

- [↑↓The SFI Quarantine Mode field is set to 01b and the actual presence detect state is deasserted \(i.e., the SFI PD State bit or the SFI OOB PD State bit transitions from Set to Clear\).↑](#)
 - [↑↓See IMPLEMENTATION NOTE: SFI QUARANTINE UPON PRESENCE DETECT STATE DEASSERTION and IMPLEMENTATION NOTE: SECURITY AND FUNCTIONAL RISKS WHEN USING IN-BAND PRESENCE DETECT for information on the use case for this mode.↑](#)
- [↑↓The SFI Quarantine Mode field is set to 10b and a\) the actual presence detect state is deasserted \(i.e., the SFI PD State bit or the SFI OOB PD State bit transitions from Set to Clear\) or b\) the Port's Data Link Layer reports DL_Down status.↑](#)
 - [↑↓See IMPLEMENTATION NOTE: SFI QUARANTINE UPON PRESENCE DETECT STATE DEASSERTION or DL_DOWN for information on the use case for this mode.↑](#)
- [↑↓The SFI Quarantine Mode field transitions to a value of 11b \(e.g., due to a Conventional Reset of Port if the default value of the SFI Quarantine Mode field is 11b or due to the intermediary modifying the SFI Quarantine Mode field to a value of 11b\).↑](#)
 - [↑↓See IMPLEMENTATION NOTE: SFI QUARANTINE AS THE DEFAULT UPON CONVENTIONAL RESET for information on the use case for this mode.↑](#)

[↑↓If the SFI Quarantine bit is Set, the following rules apply.↑](#)

- [↑↓The Port's LTSSM is immediately directed to the Disabled state until the SFI Quarantine bit is Cleared.↑](#)
- [↑↓If the Port's actual presence detect state is deasserted \(i.e., because the SFI PD State bit or the SFI OOB PD State bit transitioned from Set to Clear\), the Port masks \(i.e., Clears\) the Presence Detect State bit until the SFI Quarantine bit is Cleared.↑](#)

[↑↓Once the Port is taken out of SFI quarantine \(i.e., the SFI Quarantine bit is Cleared\), the following rules apply.↑](#)

- ↑↑↑The Port’s LTSSM must transition to the Detect State where the Link will attempt to retrain, unless otherwise specified (e.g., the Link is disabled due to another condition such as the Link Disable bit being Set).↑
 - ↑↑↑The intermediary can use the SFI DLL State bit or the SFI DLL State Changed bit to monitor when the Link reaches the DL_Active state again.↑
 - ↑↑↑Once the Link reaches the DL_Active state (i.e., the SFI DLL State bit is Set), software is permitted to modify the value of the SFI Quarantine Mode field or the SFI Quarantine bit.↑
 - ↑↑↑After re-enabling the Link, software must honor the timing requirements defined in § Section 6.6.1 with respect to the first Configuration Read following a Conventional Reset.↑
- ↑↑↑The value of the Presence Detect State bit must indicate the actual state, unless otherwise specified (e.g., the Presence Detect State bit is masked by the SFI PD State Mask bit).↑
 - ↑↑↑If the value of the Presence Detect State bit changes when the SFI Quarantine bit value changes, this must cause a Presence Detect Changed event (see § Section 6.7.3).↑

IMPLEMENTATION NOTE: SFI QUARANTINE UPON PRESENCE DETECT STATE DEASSERTION §

↑↑One of the possible use cases for this mode, enabled if the SFI Quarantine Mode field is Set to 01b, is to permit BMCs acting as the intermediary to authenticate (e.g., using CMA-SPDM) and configure (e.g., using PCIe-MI) an adapter that is hot inserted into a slot after a prior removal of an adapter from that slot.↑

↑↑Example:↑

- ↑↑Upon system boot, the BMC enables this mode on hot-plug capable Ports of slots with an adapter installed.↑
- ↑↑The system is booted into the operating system running on the host CPU(s).↑
- ↑↑If the adapter is removed, the Presence Detect State bit will be Cleared due to the removal, the Port will immediately direct the LTSSM to the Disabled state, and the Port will begin masking (i.e., Clearing) the Presence Detect State bit.↑
- ↑↑The Port will continue to direct the LTSSM to the Disabled state and will continue to mask the Presence Detect State bit until the BMC Clears the SFI Quarantine bit, ensuring that any subsequently inserted adapter is not exposed to the operating system running on the host CPU(s).↑
- ↑↑The user inserts another adapter into the slot.↑
- ↑↑The Link remains in the Disabled state, ensuring the adapter is not able to inject traffic into the platform over the Link.↑
- ↑↑Since the Link remains in the Disabled state and the Presence Detect State bit remains masked, no Data Link Layer State Changed or Presence Detect Changed event is generated to notify the operating system running on the host CPU(s) of the newly inserted adapter.↑
- ↑↑Any Configuration Read Requests attempting to pass through the quarantined Port would result in a UR Completion Status due to the Link being in the Disabled state, which hides the Adapter from any bus scans initiated by the operating system running on the host CPU(s).↑
- ↑↑While the LTSSM is in the Disabled state and the adapter is hidden from the view of the operating system running on the host CPU(s), the BMC may authenticate/configure the adapter using a sideband interface (e.g., MCTP over SMBus/I2C or MCTP over USB).↑
- ↑↑Once the BMC has authenticated/configured the adapter, the BMC Clears the SFI Quarantine bit which enables the Link and unmasks Presence Detect State in order to expose the adapter to the operating system running on the host CPU(s).↑

IMPLEMENTATION NOTE: SECURITY AND FUNCTIONAL RISKS WHEN USING IN-BAND PRESENCE DETECT §

↑↑If in-band presence detect is enabled, the presence detect state deassertion due to an adapter removal while the LTSSM is in certain states (e.g., the L1 or Disabled states) is not detected because in-band presence detect state is required to be 1b in those states, regardless of whether the adapter is present or not.↑

↑↑In these cases, if the Port is configured to direct the Link to the Disabled state upon presence detect state deassertion (i.e., the SFI Quarantine Mode field is Set to 01b), the LTSSM will not be directed to the Disabled state due to adapter removal. The BMC may be able to detect the removal and configure the Port to hide a subsequently inserted adapter from the operating system by Setting the SFI DLL State Mask bit and the SFI PD State Mask bit; however, if the subsequent insertion of another adapter occurs before the BMC has an opportunity to set these bits, the newly inserted adapter will be inadvertently exposed to the operating system running on the host CPU(s) prior to the BMC having the opportunity to authenticate/configure the newly inserted adapter. See IMPLEMENTATION NOTE: IN-BAND PRESENCE DETECT MECHANISM DEPRECATED FOR ASYNC HOT-PLUG for related functional issues.↑

↑↑To avoid this race condition for slots where the BMC is required to authenticate/configure a newly inserted adapter prior to exposing the newly inserted adapter to the operating system running on the host CPU(s), it is strongly recommended that the slot implement out-of-band presence detect, the Downstream Port implement the In-Band PD Disable bit, and in-band presence detect be disabled by Setting the In-Band PD Disable bit.↑

IMPLEMENTATION NOTE: SFI QUARANTINE UPON PRESENCE DETECT STATE DEASSERTION OR DL_DOWN §

↑↑This mode, enabled if the SFI Quarantine Mode field is Set to 10b, works similar to the mode that directs the LTSSM to the Disabled state upon presence detect state deassertion (see IMPLEMENTATION NOTE: IN-BAND PRESENCE DETECT MECHANISM DEPRECATED FOR ASYNC HOT-PLUG). But in addition to directing the LTSSM to the Disabled state upon presence detect state deassertion, this mode also directs the LTSSM to the Disabled state upon the Port's Data Link Layer reporting DL_Down status.↑

↑↑This mode is intended to handle cases where in-band presence detect is not disabled (e.g., because the slot does not support out-of-band presence detect). In this case, presence detection does not detect the removal of a device when the LTSSM is in certain states (e.g., the L1 or Disabled states) since in-band presence detect state is required to be 1b in those states. Directing the LTSSM to the Disabled state upon the Port's Data Link Layer reporting DL_Down status ensures that an adapter, inserted into a slot after the removal of another adapter in that slot, is not exposed to the operating system running on the host CPU(s) until the BMC has had an opportunity to authenticate/configure the newly inserted adapter.↑

↑↑This mode is intended to handle cases where in-band presence detect is not disabled (e.g., because the slot does not support out-of-band presence detect). In this case, presence detection does not detect the removal of a device when the LTSSM is in certain states (e.g., the L1 or Disabled states) since in-band presence detect state is required to be 1b in those states. Directing the LTSSM to the Disabled state upon the Port's Data Link Layer reporting DL_Down status ensures that an adapter, inserted into a slot after the removal of another adapter in that slot, is not exposed to the operating system running on the host CPU(s) until the BMC has had an opportunity to authenticate/configure the newly inserted adapter.↑

↑↑Note that if this mode is enabled and the operating system running on the host CPU(s) does cause a DL_Down (e.g., due to setting the Secondary Bus Reset bit or setting the Link Disable bit), after re-enabling the Link (e.g., by clearing the Secondary Bus Reset bit or clearing the Link Disable bit), the operating system waits the amount of time defined in Section 6.6.1 with respect to the first Configuration Read following a Conventional Reset. If the BMC has not re-exposed the component within this amount of time, the operating system may time out. The mechanism the BMC uses to detect the DL_Down in a timely manner is outside the scope of this specification.↑

IMPLEMENTATION NOTE: SFI QUARANTINE AS THE DEFAULT UPON CONVENTIONAL RESET §

↑↑The SFI Quarantine Mode field is HwInit allowing a Port to set the default value upon Conventional Reset to 11b. In this mode, the Port's LTSSM is directed to the Disabled state upon Conventional Reset of the Port (e.g., due to platform power on or platform reset). This mode enables a “security quarantine” of the Downstream component, preventing the component from being exposed to system software running on the host CPU(s) or being able to inject traffic into the platform over the Link until the BMC has authenticated the component and enabled the Link.↑

6.7.4.7 ~~SFI Extended Capability In-Band Hiding~~ §

ECN: Base 6.3
eSFI $\triangleleft\triangleright$

$\uparrow\downarrow$ If the SFI Hidden In-Band bit is Set, the SFI Extended Capability structure and all fields within the SFI Extended Capability structure must not be readable or writeable when accessed using Configuration Read Requests or Configuration Write Requests (e.g., initiated by in-band management system software). \uparrow

$\uparrow\downarrow$ If the SFI Hidden In-Band bit is Clear, accesses to the SFI Extended Capability follows all rules for an extended capability structure. \uparrow

$\uparrow\downarrow$ The value of the SFI Hidden In-Band bit must have no effect on accesses to the SFI Extended Capability using PCIe-MI. \uparrow

IMPLEMENTATION NOTE: USE OF SFI HIDDEN IN-BAND §

$\uparrow\downarrow$ This “hidden in-band” feature is intended for use when this SFI Extended Capability is configured by a BMC using PCIe-MI or other out-of-band management mechanisms. Setting the SFI Hidden In-Band bit hides the SFI Extended Capability when accessed by Configuration Read Requests or Configuration Write Requests (e.g., initiated by in-band management system software). \uparrow

$\uparrow\downarrow$ Hiding the SFI Extended Capability from system software running on the host CPU(s) prevents inadvertent or intentional security exploits from system software running on the host CPU(s) (e.g., by preventing system software running on the host CPU(s) from being able to modify the SFI Capability Structure to unblock a component the BMC has not authenticated or that has failed authentication). But hiding the SFI Extended Capability also prevents system software running on the host CPU(s) from being able to read the SFI Extended Capability for information that may be useful (e.g., the SFI OOB PD Supported bit). \uparrow

$\uparrow\downarrow$ If system software running on the host CPU(s) requires the ability to read the SFI Extended Capability but must be prevented from modifying the SFI Extended Capability Structure, the functionality enabled by Setting the SFI RO In-Band bit should be used instead of hiding the SFI Extended Capability. \uparrow

$\uparrow\downarrow$ The SFI Hidden In-Band bit is HwInit allowing a Port to set the default value upon Conventional Reset to 1b such that the SFI Extended Capability is always hidden when accessed by Configuration Read Requests or Configuration Write Requests. This option may be used to simplify implementations since it does not require any functionality related to SFI Extended Capability when access using Configuration Read Requests or Configuration Write Requests to be developed or validated. \uparrow

$\uparrow\downarrow$ The SFI OOB PD Supported bit may be useful to the operating system running on the host CPU(s) for configuring hot-plug related functionality and determining when an adapter is present, but that bit is not visible if the SFI Hidden In-Band bit is Set. In this case, the SCap2 OOB PD Supported bit (if implemented) can be used by software instead. \uparrow

$\uparrow\downarrow$ For systems that support firmware first DPC, the SFI HPS Suppress bit supports the use case defined by the DSM for Downstream Port Containment and Hot-Plug Surprise Control in the PCI Firmware Specification. If the SFI Extended Capability is hidden on such systems when accessed by Configuration Read Requests or Configuration Write Requests, mechanisms outside the scope of this specification may be required to support this use case. \uparrow

ECN: Base 6.3
eSFI $\Delta\triangleleft\triangleright$

6.7.4.8 $\uparrow\downarrow$ SFI Extended Capability In-Band Read-Only \S

- $\uparrow\downarrow$ If the SFI RO In-Band [hotlink] bit is Set, Configuration Read Requests (e.g., initiated by in-band management [hotlink] system software) to the SFI Extended Capability [hotlink] are processed with normal semantics, except Configuration Read Requests to the SFI CAM Data [hotlink] register which must return a value of 0h without generating a Configuration Read Request on the Link below the Port. If the SFI RO In-Band [hotlink] bit is Set and a Configuration Write Request accesses the SFI CAM Data [hotlink] register, the value of the SFI CAM Data [hotlink] register must not be altered and a Configuration Write Request must not be generated on the Link below the Port. \uparrow
- $\uparrow\downarrow$ If the SFI RO In-Band [hotlink] bit is Clear, Configuration Read Requests and Configuration Write Requests to the SFI Extended Capability must be processed with normal semantics, unless otherwise specified (e.g., the SFI CAM Unsupported bit is Set). \uparrow
- $\uparrow\downarrow$ The value of the SFI RO In-Band [hotlink] bit must have no effect on accesses to the SFI Extended Capability [hotlink] using PCIe-MI [hotlink]. \uparrow

IMPLEMENTATION NOTE: USE OF SFI RO IN-BAND \S

- $\uparrow\downarrow$ This “read-only in-band” feature is intended for use when the SFI Extended Capability is configured by a BMC using PCIe-MI [hotlink] or other out-of-band management [hotlink] mechanisms. Setting the SFI RO In-Band [hotlink] bit blocks system software running on the host CPU(s) from writing to the SFI Extended Capability [hotlink]. Blocking such Configuration Write Requests prevents inadvertent or intentional security exploits from system software running on the host CPU(s) (e.g., by preventing the ability to modify the SFI Capability Structure to unblock access to an unauthenticated component) while still allowing Configuration Read Requests so that system software running on the host CPU(s) is able to read the SFI Extended Capability [hotlink] for information that may be useful (e.g., the SFI OOB PD Supported bit). \uparrow
- $\uparrow\downarrow$ The SFI RO In-Band [hotlink] bit is HwInit [hotlink] allowing a Port to set the default value upon Conventional Reset to 1b such that the SFI Extended Capability [hotlink] is always read-only when accessed by Configuration Read Requests or Configuration Write Requests. This option may be useful to avoid race cases where system software running on the host CPUs may be able to write to the SFI Extended Capability after a Conventional Reset and before the BMC has an opportunity to set the SFI RO In-Band [hotlink] bit. \uparrow
- $\uparrow\downarrow$ For systems that support firmware first DPC, the SFI HPS Suppress [hotlink] bit supports the use case defined by the DSM for Downstream Port Containment and Hot-Plug Surprise Control in the PCI Firmware Specification. If the SFI Extended Capability [hotlink] is read-only on such systems when accessed by Configuration Read Requests or Configuration Write Requests, mechanisms outside the scope of this specification may be required to support this use case. \uparrow

6.7.5 Firmware Support for Hot-Plug \S

Some systems that include hot-plug capable Root Ports and Switches that are released before ACPI-compliant operating systems with native hot-plug support are available, can use ACPI firmware for propagating hot-plug events. Firmware control of the hot-plug registers must be disabled if an operating system with native support is used. Platforms that provide ACPI firmware to propagate hot-plug events must also provide a mechanism to transfer control to the operating system. The details of this method are described in the *PCI Firmware Specification*.

6.7.6 Async Removal §

Async removal refers to the removal of an adapter or disabling of a Downstream Port Link due to error containment without prior warning to the operating system. This is in contrast to orderly removal, where removal operations are performed in a lock-step manner with the operating system through a well defined sequence of user actions and system management facilities. For example, the user presses the Attention Button to request permission from the operating system to remove the adapter, but the user doesn't actually remove the adapter from the slot until the operating system has quiesced activity to the adapter and granted permission for removal.

Since async removal proceeds before the rest of the PCI Express hierarchy or operating system necessarily becomes aware of the event, special consideration is required beyond that needed for standard PCI hot-plug. This section outlines PCI Express events that may occur as a side effect of async removal and mechanisms for handling async removal.

Since async removal may be unexpected to both the Physical and Data Link Layers of the Downstream Port associated with the slot, Correctable Errors may be reported as a side effect of the event (i.e., Receiver Error, Bad TLP, and Bad DLLP). If these errors are reported, software should handle them as an expected part of this event.

Requesters may experience Completion Timeouts associated with Requests that were accepted, but will never be completed by removed Completers. Any resulting Completion Timeout errors in this context should be handled as an expected part of this event.

Async removal may result in a transition from DL_Active to DL_Down in the Downstream Port. This transition may result in a Surprise Down error. In addition, Requesters in the PCI Express hierarchy domain may not become immediately aware of this transition and continue to issue Requests to removed Completers that must be handled by the Downstream Port associated with the slot.

Either Downstream Port Containment (DPC) or the Hot-Plug Surprise (HPS) mechanism may be used to support async removal as part of an overall async hot-plug architecture. See § Appendix I. for the associated reference model.

IMPLEMENTATION NOTE: HOT-PLUG SURPRISE MECHANISM DEPRECATED FOR ASYNC HOT-PLUG §

The Hot-Plug Surprise (HPS) mechanism, as indicated by the Hot-Plug Surprise bit in the Slot Capabilities Register being Set, is deprecated for use with async hot-plug. DPC is the recommended mechanism for supporting async hot-plug. See § Section 6.7.4.4 for guidance on slots supporting both mechanisms.

With async removal, using HPS has serious downsides. Uncorrectable errors other than those that inherently bring down the Link need to be configured either to crash the system, be handled asynchronously by software, or be ignored. These include uncorrectable errors associated with Posted Memory Writes, TLPs with poisoned data, and Completion Timeouts. Uncorrectable errors ignored or handled asynchronously by software may make it impossible for the driver to determine which high-level operations complete successfully versus those that do not.

DPC provides a robust mechanism for supporting async removal. The TLP stream cleanly stops upon an uncorrectable error that triggers DPC. Operating System / driver stacks that support Containment Error Recovery (CER) can fully and transparently recover from many transient PCIe uncorrectable errors. DPC can support async removal and CER concurrently

6.8 Power Budgeting Mechanism §

With the addition of a hot-plug capability for adapters, the need arises for the system to be capable of properly allocating power to any new devices added to the system. This capability is a separate and distinct function from power management and a basic level of support is required to ensure proper operation of the system. The power budgeting concept puts in place the building blocks that allow devices to interact with systems to achieve these goals. There are many ways in which the system can implement the actual power budgeting capabilities, and as such, they are beyond the scope of this specification.

Implementation of the Power Budgeting Extended Capability is optional for devices that are implemented either in a form factor that does not require hot-plug support, or that are integrated on the system board. Form factor specifications may require support for power budgeting. The devices and/or adapters are required to remain under the configuration power limit specified in the corresponding electromechanical specification until they have been configured and enabled by the system. The system should guarantee that power has been properly budgeted prior to enabling an adapter.

When enabled, Extended Power Budgeting provides power consumption information on a per-connector basis. This allows a system to manage power consumption more accurately.

§ Table 6-10 shows the deployment mechanisms for Power Budgeting.

Table 6-10 Power Budgeting Deployments §

| Device Type | Power Budgeting Extended Capability | Description | Notes |
|--|--|---|--|
| Single Function, Multi-Function Device , or Multi-Device add-in card | Not Present | No Power Budgeting information is available | Some form factors may not permit this combination |
| Single-Function Device | Present | Power Budgeting represents Device power | |
| Multi-Function Device | Present in exactly one Function | | |
| Multi-Function Device | Present in more than one Function | Power Budgeting represents per-Function power | <p>Sum of all Functions represents power consumed by the Device.</p> <p>Functions that do not implement Power Budgeting have negligible power consumption.</p> <p>Power that is not associated with a specific Function is included in an implementation specific manner.</p> |
| Multi-Device add-in card | Present in one or more Functions of exactly one Device | Power Budgeting represents add-in card power | <p>Sum of all Functions represents power consumed by the add-in card.</p> <p>Functions and Devices that do not implement Power Budgeting have negligible power consumption.</p> <p>Power that is not associated with a specific Function is included in an implementation specific manner.</p> |

| Device Type | Power Budgeting Extended Capability | Description | Notes |
|--------------------------|-------------------------------------|---|--|
| Multi-Device add-in card | Present in more than one Device | Power Budgeting represents per-Device power | <p>Sum of all Devices represents power consumed by the add-in card.</p> <p>Devices that do not implement Power Budgeting have negligible power consumption.</p> <p>Power that is not associated with a specific Device is included in an implementation specific manner.</p> |

6.8.1 System Power Budgeting Process Recommendations §

It is recommended that system firmware provide the power budget management agent the following information:

- Total system power budget (power supply information).
- Total power allocated by system firmware (system board devices).
- Total number of slots and the types of slots.

System firmware is responsible for allocating power for all devices on the system board that do not have power budgeting capabilities. The firmware may or may not include devices that are connected to the standard power rails. When the firmware allocates the power for a device that implements the Power Budgeting Extended Capability it must set the System Allocated bit to 1b in the Power Budget Capability register to indicate that it has been properly allocated. The power budget manager is responsible for allocating all PCI Express devices including system board devices that have the Power Budgeting Extended Capability and have the System Allocated bit Clear. The power budget manager is responsible for determining if hot-plugged devices can be budgeted and enabled in the system.

There are alternate methods which may provide the same functionality, and it is not required that the power budgeting process be implemented in this manner.

6.8.2 Device Power Considerations §

When the Power Budgeting Extended Capability is present in more than one Function of a Device (or the only Function of a Single-Function Device), power is reported on a per-Function basis (see § Section 7.8.1). When Power Budgeting is present in exactly one Function of a Multi-Function Device, power is reported on a per-Device basis. When Power Budgeting is present in more than one Function, but not in all Functions, power is reported on a per-Function basis and the missing Functions are treated as consuming negligible power.

When Power Budgeting is present in exactly one Function, this represents per-Device Power. When Power Budgeting is present in exactly one Device of a Multi-Device add-in card, power is reported on a per-add-in card basis. The following rules apply:

- When all Functions in the Device are in the same PM State, power budgeting for that PM State applies.
 - If no power budgeting is reported for a given PM State, the next higher PM State that is reported applies.
- When Functions are in different PM States, power budgeting is at an implementation specific value between the power budgeting reported for highest and lowest power PM State (e.g., if no Functions are in D0, one

Function is in D1 and another Function is in D3, power budgeting is somewhere between the values reported for D1 and D3 , inclusive).

- If no power budgeting is reported for the highest PM State, power budgeting for the next higher PM State that is reported applies.
- If no power budgeting is reported for the lowest PM State, power budgeting for the next lower PM State that is reported applies.

6.8.3 Power Limit Mechanisms §

An add-in card must not consume more power than it is granted by the system. There are six mechanisms defined to grant power to an add-in card:

1. **Initial Power** – this is power granted to all similarly situated add-in cards by the form factor. This value is form factor specific and typically is based on factors like add-in card size and external power configuration.
2. **Set_Slot_Power_Limit power** – this is power granted to an add-in card when it receives a Set_Slot_Power_Limit message with a power value greater than the Initial Power grant for the add-in card.
3. **Power Limit PM Sub State power** – this is power granted to an add-in card through the Power Limit mechanism (see below). This mechanism overrides power grants from either the Initial Power or Set_Slot_Power_Limit mechanisms. Unlike Set_Slot_Power_Limit , this mechanism is permitted to grant a power level that is lower than the Initial Power grant.
4. **Out of Band Power Limit PM Sub State power** – this is power granted to an add-in card through the Out of Band Power Limit mechanism (see below). This mechanism overrides power grants from either the Initial Power or Set_Slot_Power_Limit mechanisms. Unlike Set_Slot_Power_Limit , this mechanism is permitted to grant a power level that is lower than the Initial Power grant.
5. **Firmware based additional Aux Power** – this is Aux power granted when the driver uses the Request D3 Cold Aux Power Limit _DSM call as defined in [Firmware]. This mechanism overrides Aux power grants using any of the above mechanisms.
6. **Form factor specific or Device Class (specific e.g., NVMe) mechanisms** – this is power granted to an add-in card using mechanisms defined by other specifications and are outside the scope of this specification. Interactions between these mechanisms and the mechanisms listed above are form factor or Device Class specific.

The Power Limit mechanism is optional. Support is indicated by the Power Limit Supported bit in the lowest-numbered Function that contains a Power Budgeting Extended Capability . The Power Limit fields in that Function control power for the entire add-in card.

The Power Budgeting Data Register (see § Section 7.8.1.3) contains a PM Sub State field. This permits a Device to report power consumption for up to 8 PM Sub States. This PM Sub State is implementation specific and is unrelated to other similarly named concepts in PCIe (e.g., PM State, L1 PM Substates).

Each PM Sub State represents additional optional power consumption levels. Each PM Sub State is mutually exclusive. At any specific point in time, a Device operates in exactly one PM Sub State. Each defined power consumption level includes a full complement of Data entries (e.g., Aux Power, Thermal, Max, Min for all appropriate connectors and power rails).

Device behavior is undefined if software attempts to transition the Device to a PM Sub State for which there are no Data entries.

When both the Power Limit Enable and the Out of Band Power Limit Enable bits are Clear, for each power rail and connector combination, the Device must operate in the PM Sub State determined by an implementation specific mechanism.

When both the Power Limit Enable and Out of Band Power Limit Enable bits are Set, individually for each power rail and connector combination, the Device must operate in the PM Sub State indicated by the smaller power consumption indicated by Power Limit PM Sub State or Out of Band Power Limit PM Sub State fields.

The Power Limit Enable and Power Limit PM Sub State fields are configured using Configuration Write transactions while all Functions of the Device are in D0 uninitialized . Behavior is undefined if these fields change after any Function exits D0uninitialized .

The Out of Band Power Limit Enable and Out of Band Power Limit PM Sub State values are configured using implementation specific mechanisms. The mechanism used for out of band configuration is outside the scope of this specification.

6.9 Slot Power Limit Control §

PCI Express provides a mechanism for software controlled limiting of the Maximum Power per slot that an adapter (associated with that slot) can consume. If supported, the Emergency Power Reduction State, over-rides the mechanisms listed here (see § Section 6.24). The key elements of this mechanism are:

- Slot Power Limit Value and Scale fields of the Slot Capabilities register implemented in the Downstream Ports of a Root Complex or a Switch
- Captured Slot Power Limit Value and Scale fields of the Device Capabilities register implemented in Endpoint, Switch, or PCI Express-PCI Bridge Functions present in an Upstream Port
- Set_Slot_Power_Limit Message that conveys the content of the Slot Power Limit Value and Scale fields of the Slot Capabilities register of the Downstream Port (of a Root Complex or a Switch) to the corresponding Captured Slot Power Limit Value and Scale fields of the Device Capabilities register in the Upstream Port of the component connected to the same Link

Power limits on the platform are typically controlled by the software (for example, platform firmware) that comprehends the specifics of the platform such as:

- Partitioning of the platform, including slots for I/O expansion using adapters
- Power delivery capabilities
- Thermal capabilities

This software is responsible for correctly programming the Slot Power Limit Value and Scale fields of the Slot Capabilities registers of the Downstream Ports connected to slots. After the value has been written into the register within the Downstream Port, it is conveyed to the adapter using the Set_Slot_Power_Limit Message (see § Section 2.2.8.5). The recipient of the Message must use the value in the Message data payload to limit usage of the power for the entire adapter, unless the adapter will never exceed the lowest value specified in the corresponding form factor specification. It is required that device driver software associated with the adapter be able (by reading the values of the Captured Slot Power Limit Value and Scale fields of the Device Capabilities register) to configure hardware of the adapter to guarantee that the adapter will not exceed the imposed limit. In the case where the platform imposes a limit that is below the minimum needed for adequate operation, the device driver will be able to communicate this discrepancy to higher level configuration software. Configuration software is required to set the Slot Power Limit to one of the maximum values specified for the corresponding form factor based on the capability of the platform.

The following rules cover the Slot Power Limit control mechanism:

For Adapters:

- Until and unless a Set_Slot_Power_Limit Message is received indicating a Slot Power Limit value greater than the lowest value specified in the form factor specification for the adapter's form factor, the adapter must not consume more than the lowest value specified.
- An adapter must never consume more power than what was specified in the most recently received Set_Slot_Power_Limit Message or the minimum value specified in the corresponding form factor specification, whichever is higher.
- Components with Endpoint, Switch, or PCI Express-PCI Bridge Functions that are targeted for integration on an adapter where total consumed power is below the lowest limit defined for the targeted form factor are permitted to ignore Set_Slot_Power_Limit Messages , and to return a value of 0 in the Captured Slot Power Limit Value and Scale fields of the Device Capabilities register
 - Such components still must be able to receive the Set_Slot_Power_Limit Message without error but simply discard the Message value

For Root Complex and Switches which source slots:

- Configuration software must not program a Set_Slot_Power_Limit value that indicates a limit that is lower than the lowest value specified in the form factor specification for the slot's form factor.

IMPLEMENTATION NOTE: EXAMPLE ADAPTER BEHAVIOR BASED ON THE SLOT POWER LIMIT CONTROL CAPABILITY §

The following power limit scenarios are examples of how an adapter must behave based on the Slot Power Limit control capability. The form factor limits are representations, and should not be taken as actual requirements.

Note: Form factor #1 has a Maximum Power requirement of 40 W and 25 W; form factor #2 has a Maximum Power requirement of 15 W.

Scenario 1: An Adapter Consuming 12 W

- If the adapter is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.

In all cases, since the adapter operates normally within all the form factors, it can ignore any of the slot power limit Messages.

Scenario 2: An Adapter Consuming 18 W

- If the adapter is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the adapter must scale down to 15 W or disable operation. An adapter that does not scale within any of the power limits for a given form factor will always be disabled in that form factor and should not be used.

In this case, if the adapter is only to be used in form factor #1, it can ignore any of the slot power limit Messages. To be useful in form factor #2, the adapter should be capable of scaling to the power limit of form factor #2.

Scenario 3: An Adapter Consuming 30 W

- If the adapter is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.
- If the adapter is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the device must scale down to 25 W or disable operation.
- If the adapter is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the adapter must scale down to 15 W or disable operation. An adapter that does not scale within any of the power limits for a given form factor will always be disabled in that form factor and should not be used.

In this case, since the adapter consumes power above the lowest power limit for a slot, the adapter must be capable of scaling or disabling to prevent system failures. Operation of adapters at power levels that exceed the capabilities of the slots in which they are plugged must be avoided.

IMPLEMENTATION NOTE: SLOT POWER LIMIT CONTROL REGISTERS §

Typically Slot Power Limit register fields within Downstream Ports of a Root Complex or a Switch will be programmed by platform-specific software. Some implementations may use a hardware method for initializing the values of these registers and, therefore, do not require software support.

Components with Endpoint, Switch, or PCI Express-PCI Bridge Functions that are targeted for integration on the adapter where total consumed power is below the lowest limit defined for that form factor are allowed to ignore Set_Slot_Power_Limit Messages. Note that components that take this implementation approach may not be compatible with potential future defined form factors. Such form factors may impose lower power limits that are below the minimum required by a new adapter based on the existing component.

IMPLEMENTATION NOTE: AUTO SLOT POWER LIMIT DISABLE §

In some environments host software may wish to directly manage the transmission of a Set_Slot_Power_Limit message by performing a Configuration Write to the Slot Capabilities register rather than have the transmission automatically occur when the Link transitions from a non- DL_Up to a DL_Up status. This allows host software to limit power supply surge current by staggering the transition of Endpoints to a higher power state following a Link Down or when multiple Endpoints are simultaneously hot-added due to cable or adapter insertion.

6.10 Root Complex Topology Discovery §

A Root Complex may present one of the following topologies to configuration software:

- A single opaque Root Complex such that software has no visibility with respect to internal operation of the Root Complex. All Root Ports are independent of each other from a software perspective; no mechanism exists to manage any arbitration among the various Root Ports for any differentiated services.
- A single Root Complex Component such that software has visibility and control with respect to internal operation of the Root Complex Component. As shown in § Figure 6-11 , software views the Root Ports as Ingress Ports for the component. The Root Complex internal Port for traffic aggregation to a system Egress Port or an internal sink unit (such as memory) is represented by an RCRB structure. Controls for differentiated services are provided through a Virtual Channel Capability structure located in the RCRB .

Base 6.4 vs Base 6.3

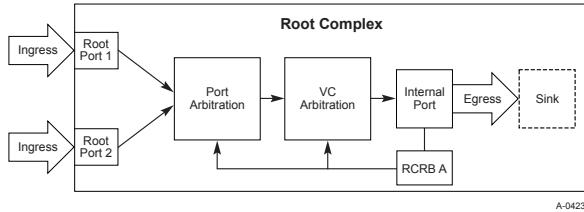


Figure 6-11 Root Complex Represented as a Single Component §

- Multiple Root Complex Components such that software not only has visibility and control with respect to internal operation of a given Root Complex Component but also has the ability to discover and control arbitration between different Root Complex Components. As shown in § Figure 6-12 , software views the Root Ports as Ingress Ports for a given component. An RCRB structure controls egress from the component to other Root Complex Components (RCRB C) or to an internal sink unit such as memory (RCRB A). In addition, an RCRB structure (RCRB B) may also be present in a given component to control traffic from other Root Complex Components. Controls for differentiated services are provided through Virtual Channel Capability structures located appropriately in the RCRBs respectively.

More complex topologies are possible as well.

A Root Complex topology can be represented as a collection of logical Root Complex Components such that each logical component has:

- One or more Ingress Ports.
- An Egress Port.
- Optional associated Virtual Channel capabilities located either in the Configuration Space (for Root Ports) or in an RCRB (for internal Ingress/Egress Ports) if the Root Complex supports Virtual Channels.
- Optional devices/Functions integrated in the Root Complex.

In order for software to correctly program arbitration and other control parameters for PCI Express differentiated services, software must be able to discover a Root Complex's internal topology. Root Complex topology discovery is accomplished by means of the Root Complex Link Declaration Capability as described in § Section 7.9.8 .

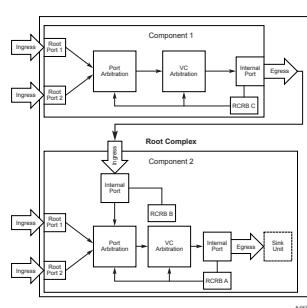


Figure 6-12 Root Complex Represented as Multiple Components §

6.11 Link Speed Management §

This section describes how Link speed management is coordinated between the LTSSM (§ Section 4.2.7) and the software Link observation and control mechanisms (see § Section 7.5.3.6 , § Section 7.5.3.7 , § Section 7.5.3.8 , § Section 7.5.3.18 , § Section 7.5.3.19 , and § Section 7.5.3.20).

The Target Link Speed field in the Link Control 2 register in the Downstream Port sets the upper bound for the Link speed. Except as described below, the Upstream component must attempt to maintain the Link at the Target Link Speed, or at the highest speed supported by both components on the Link (as reported by the values in the training sets - see § Section 4.2.5.1), whichever is lower.

Any Upstream Port or Downstream Port with the Hardware Autonomous Speed Disable bit in the Link Control 2 register clear is permitted to autonomously change the Link speed using implementation specific criteria.

If the reliability of the Link is unacceptably low, then either component is permitted to lower the Link speed by removing the unreliable Link speed from the list of supported speeds advertised in the training sets the component transmits. The criteria for determination of acceptable Link reliability are implementation specific, and are not dependent on the setting of the Hardware Autonomous Speed Disable bit.

During any given speed negotiation it is possible that one or both components will advertise a subset of all speeds supported, as a means to cap the post-negotiation Link speed. It is permitted for a component to change its set of advertised supported speeds without requesting a Link speed change by driving the Link through Recovery without setting the speed change bit.

When a component's attempt to negotiate to a particular Link speed fails, that component is not permitted to attempt negotiation to that Link speed, or to any higher Link speed, until 200 ms has passed from the return to L0 following the failed attempt, or until the other component on the Link advertises support for the higher Link speed through its transmitted training sets (with or without a request to change the Link speed), whichever comes first.

Software is permitted to restrict the maximum speed of Link operation and set the preferred Link speed by setting the value in the Target Link Speed field in the Upstream component. After modifying the value in the Target Link Speed field, software must trigger Link retraining by writing 1b to the Retrain Link bit. Software is notified of any Link speed changes (as well as any Link width changes) through the Link Bandwidth Notification Mechanism.

Software is permitted to cause a Link to transition to the Polling.Compliance LTSSM state at a particular speed by writing the Link Control 2 register in both components with the same value in the Target Link Speed field and Setting the Enter Compliance bit, and then initiating a Hot Reset on the Link (through the Downstream Port).

Note that this will take the Link to a DL_Down state and therefore cannot be done transparently to other software that is using the Link. The Downstream Port will return to Polling.Active when the Enter Compliance bit is cleared.

6.12 Access Control Services (ACS) §

ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected. ACS is applicable to RCs, Switches, and Multi-Function Devices .¹³⁹ For ACS requirements, Single-Function Devices that are SR-IOV capable ↑or SIOV capable must be handled as if they were Multi-Function Devices , since they essentially behave as Multi-Function Devices after their Virtual Functions (VFs) ↑or Scalable Device Interfaces (SDIs)↑ are enabled.

ECN: Base 6.3
SIOV△↔

Implementation of ACS in RCiEPs is permitted but not required. It is explicitly permitted that, within a single Root Complex, some RCiEPs implement ACS and some do not. It is strongly recommended that Root Complex

¹³⁹. Applicable Functions within Multi-Function Devices specifically include PCI Express Endpoints, Switch Upstream Ports, Legacy PCI Express Endpoints, and Root Complex Integrated Endpoints.

implementations ensure that all accesses originating from RCiEPs $\downarrow\downarrow(PFs\downarrow\uparrow(PFs, VFs,\uparrow)$ and $\uparrow\downarrow(VFs)\downarrow\uparrow\downarrow(SDIs)\uparrow$ without ACS $\downarrow\downarrow(capability\downarrow\uparrow(support\uparrow)$ are first subjected to processing by the Translation Agent (TA) in the Root Complex before further decoding and processing. The details of such Root Complex handling are outside the scope of this specification.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

ACS provides the following types of access control:

- ACS Source Validation
- ACS Translation Blocking
- ACS P2P Request Redirect
- ACS P2P Completion Redirect
- ACS Upstream Forwarding
- ACS P2P Egress Control
- ACS Direct Translated P2P
- ACS I/O Request Blocking
- ACS DSP Memory Target Access
- ACS USP Memory Target Access
- ACS Unclaimed Request Redirect

The specific requirements for each of these are discussed in the following section.

ACS hardware functionality is disabled by default, and is enabled only by ACS-aware software. With the exception of ACS Source Validation, ACS access controls are not applicable to Multicast TLPs (see § Section 6.14), and have no effect on them.

6.12.1 ACS Component Capability Requirements §

ACS functionality is reported and managed via ACS Extended Capability structures. PCI Express components are permitted to implement ACS Extended Capability structures in some, none, or all of their applicable Functions. The extent of what is implemented is communicated through capability bits in each ACS Extended Capability structure. A given Function with an ACS Extended Capability structure may be required or forbidden to implement certain capabilities, depending upon the specific type of the Function and whether it is part of a Multi-Function Device.

ACS is never applicable to a PCI Express to PCI Bridge Function or a Root Complex Event Collector Function, and such Functions must never implement an ACS Extended Capability structure.

6.12.1.1 ACS Downstream Ports §

This section applies to Root Ports and Switch Downstream Ports that implement an ACS Extended Capability structure. This section applies to Downstream Port Functions both for Single-Function Devices and Multi-Function Devices.

- ACS Source Validation : must be implemented.

When enabled, the Downstream Port tests the Bus Number from the Requester ID of each Upstream Request received by the Port to determine if it is associated with the Secondary side of the virtual bridge associated with the Downstream Port, by either or both of:

- Determining that the Requester ID falls within the Bus Number “aperture” of the Port - the inclusive range specified by the ~~↓↑Secondary Bus Number register~~ ↑↓Secondary Bus Number Register and the ~~↓↑Subordinate Bus Number register~~ ↑↓Subordinate Bus Number Register.
- If FPB is implemented and enabled, determining that the Requester ID is associated with the bridge’s Secondary Side by the application of the FPB Routing ID mechanism.

If the Bus Number from the Requester ID of the Request is not within this aperture, this is a reported error (ACS Violation) associated with the Receiving Port (see § Section 6.12.5.)

ACS mechanisms are not architected to check Segment Numbers in FM TLPs. Segment Number checking that applies independently of ACS is specified under Segment Rules in § Section 2.2.1.2.

Completions are never affected by ACS Source Validation.

IMPLEMENTATION NOTE: UPSTREAM MESSAGES AND ACS SOURCE VALIDATION §

Functions are permitted to transmit Upstream Messages before they have been assigned a Bus Number. Such messages will have a Requester ID with a Bus Number of 00h. If the Downstream Port has ACS Source Validation enabled, these Messages (see § Table F-1, § Section 2.2.8.2, and § Section 6.22.1) will likely be detected as an ACS Violation error.

- ACS Translation Blocking: must be implemented.

When enabled, the Downstream Port checks the Address Type (AT) field of each Upstream Memory Request and, in Flit ~~↓↑mode~~ ↑↓Mode only, of each Upstream Address Routed Message received by the Port. If the AT field is not 00b (Untranslated), this is a reported error (ACS Violation) associated with the Receiving Port (see § Section 6.12.5). This error must take precedence over ACS Upstream Forwarding and any applicable ACS P2P control mechanisms.

Completions are never affected by ACS Translation Blocking.

- ACS P2P Request Redirect: must be implemented by Root Ports that support peer-to-peer traffic with other Root Ports; ¹⁴⁰ must be implemented by Switch Downstream Ports.

ACS P2P Request Redirect is subject to interaction with the ACS P2P Egress Control and ACS Direct Translated P2P mechanisms (if implemented). Refer to § Section 6.12.3 for more information.

When ACS P2P Request Redirect is enabled in a Switch Downstream Port, peer-to-peer Requests must be redirected Upstream towards the RC.

When ACS P2P Request Redirect is enabled in a Root Port, peer-to-peer Requests must be sent to Redirected Request Validation logic within the RC that determines whether the Request is “reflected” back Downstream towards its original target, or blocked as an ACS Violation error. The algorithms and specific controls for making this determination are not architected by this specification.

Downstream Ports never redirect Requests that are traveling Downstream.

Completions are never affected by ACS P2P Request Redirect.

- ACS P2P Completion Redirect: must be implemented by Root Ports that implement ACS P2P Request Redirect; must be implemented by Switch Downstream Ports.

¹⁴⁰ 40. Root Port indication of ACS P2P Request Redirect or ACS P2P Completion Redirect support does not imply any particular level of peer-to-peer support by the Root Complex, or that peer-to-peer traffic is supported at all.

The intent of ACS P2P Completion Redirect is to avoid ordering rule violations between Completions and Requests when Requests are redirected. Refer to § [Section 6.12.6](#) for more information.

ACS P2P Completion Redirect does not interact with ACS controls that govern Requests.

When ACS P2P Completion Redirect is enabled in a Switch Downstream Port, peer-to-peer Completions¹⁴¹ that do not have the Relaxed Ordering Attribute bit set (1b) must be redirected Upstream towards the RC.

Otherwise, peer-to-peer Completions must ~~↑↑not↑~~ be ~~↑↓routed normally.↓~~
~~↑↑redirected.↑~~

Errata: Base 6.3
B836△◀▶

When ACS P2P Completion Redirect is enabled in a Root Port, peer-to-peer Completions that do not have the Relaxed Ordering bit set must be handled such that they do not pass Requests that are sent to Redirected Request Validation logic within the RC. Such Completions must eventually be sent Downstream towards their original peer-to-peer targets, without incurring additional ACS access control checks.

Downstream Ports never redirect Completions that are traveling Downstream.

Requests are never affected by ACS P2P Completion Redirect.

- ACS Upstream Forwarding: must be implemented by Root Ports if the RC supports Redirected Request Validation; must be implemented by Switch Downstream Ports.

When ACS Upstream Forwarding is enabled in a Switch Downstream Port, and its Ingress Port receives an Upstream Request or Completion TLP targeting the Port's own Egress Port, the Port must instead forward the TLP Upstream towards the RC.

When ACS Upstream Forwarding is enabled in a Root Port, and its Ingress Port receives an Upstream Request or Completion TLP that targets the Port's own Egress Port, the Port must handle the TLP as follows. For a Request, the Root Port must handle it the same as a Request that the Port “redirects” with the ACS P2P Request Redirect mechanism. For a Completion, the Root Port must handle it the same as a Completion that the Port “redirects” with the ACS P2P Completion Redirect mechanism.

When ACS Upstream Forwarding is not enabled on a Downstream Port, and its Ingress Port receives an Upstream Request or Completion TLP that targets the Port's own Egress Port, the handling of the TLP is undefined.

- ACS P2P Egress Control: implementation is optional.
ACS P2P Egress Control is subject to interaction with the ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms (if implemented). Refer to § [Section 6.12.3](#) for more information.

A Switch that supports ACS P2P Egress Control can be selectively configured to block peer-to-peer Requests between its Downstream Ports. Software can configure the Switch to allow none or only a subset of its Downstream Ports to send peer-to-peer Requests to other Downstream Ports. This is configured on a per Downstream Port basis.

An RC that supports ACS P2P Egress Control can be selectively configured to block peer-to-peer Requests between its Root Ports. Software can configure the RC to allow none or only a subset of the Hierarchy Domains to send peer-to-peer Requests to other Hierarchy Domains. This is configured on a per Root Port basis.

With ACS P2P Egress Control in Downstream Ports, controls in the Ingress Port (“sending” Port) determine if the peer-to-peer Request is blocked, and if so, the Ingress Port handles the ACS Violation error per § [Section 6.12.5](#).

Completions are never affected by ACS P2P Egress Control.

141. This includes Read Completions, AtomicOp Completions, and other Completions with or without Data.

- ACS Direct Translated P2P: must be implemented by Root Ports that support Address Translation Services (ATS) and also support peer-to-peer traffic with other Root Ports;¹⁴² must be implemented by Switch Downstream Ports.

When ACS Direct Translated P2P is enabled in a Downstream Port, peer-to-peer Memory Requests and, in Flit ~~↓↓mode~~ ↑↑Mode only, peer-to-peer Address Routed Messages whose Address Type (AT) field indicates a Translated address must be routed ~~↓↓normally ("directly")~~ to the peer Egress ~~↓↓Port~~, ↑↑Port without redirection, regardless of ACS P2P Request Redirect and ACS P2P Egress Control settings. All other peer-to-peer Requests must still be subject to ACS P2P Request Redirect and ACS P2P Egress Control settings.

Errata: Base 6.3
B836△◀▷

Completions are never affected by ACS Direct Translated P2P.

- ACS I/O Request Blocking : must be implemented by Root Ports and Switch Downstream Ports that support ACS Enhanced Capability.

When enabled, the Port must handle an Upstream I/O Request received by the Port's Ingress as an ACS Violation.

- ACS DSP Memory Target Access : must be implemented by Root Ports and Switch Downstream Ports that support ACS Enhanced Capability and that have applicable Memory BAR Space to protect.

ACS DSP Memory Target Access determines how an Upstream Request received by the Downstream Port's Ingress and targeting any Memory BAR Space¹⁴³ associated with an applicable Downstream Port is handled. The Request can be blocked, redirected, or allowed to proceed directly to its target. In a Switch, all Downstream Ports are applicable, including the one on which the Request was received. In a Root Complex, the set of applicable Root Ports is implementation specific, but always includes the one on which the Request was received.

- ACS USP Memory Target Access : must be implemented by Switch Downstream Ports that support ACS Enhanced Capability and that have applicable Memory BAR Space in the Switch Upstream Port to protect; is not applicable to Root Ports.

ACS USP Memory Target Access determines how an Upstream Request received by the Switch Downstream Port's Ingress and targeting any Memory BAR Space¹⁴⁴ associated with the Switch's Upstream Port is handled. The Request can be blocked, redirected, or allowed to proceed directly to its target.

If any Functions other than the Switch Upstream Port are associated with the Upstream Port, this field has no effect on accesses to their Memory BAR Space¹⁴⁵. Such access is controlled by the ACS Extended Capability (if present) in the Switch Upstream Port.

- ACS Unclaimed Request Redirect : must be implemented by Switch Downstream Ports that support ACS Enhanced Capability; is not applicable to Root Ports.

When enabled, incoming Requests received by the Switch Downstream Port's Ingress and targeting Memory Space within the memory window of a Switch Upstream Port that is not within a memory window or Memory BAR Target of any Downstream Port within the Switch are redirected Upstream out of the Switch.

When not enabled, such Requests are handled by the Switch Downstream Port as an Unsupported Request (UR).

142. Root Port indication of ACS Direct Translated P2P support does not imply any particular level of peer-to-peer support by the Root Complex, or that peer-to-peer traffic is supported at all.

143. This also includes any Memory Space allocated by an Expansion ROM Base Address register (BAR). This also includes any Memory Space allocated by EA entries with a BEI value of 0, 1, 7, or 8. See § Section 7.8.5.3.

144. This also includes any Memory Space allocated by an Expansion ROM Base Address register (BAR). This also includes any Memory Space allocated by EA entries with a BEI value of 0, 1, 7, or 8. See § Section 7.8.5.3.

145. This also includes any Memory Space allocated by an Expansion ROM Base Address register (BAR). This also includes any Memory Space allocated by EA entries with a BEI value of 0, 1, 7, or 8. See § Section 7.8.5.3.

6.12.1.2 ACS Functions in ~~Not SR-IOV Capable~~ SR-IOV, SIOV, and Multi-Function Devices §

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

This section applies to Multi-Function Device ACS Functions, with the exception of Downstream Port Functions, which are covered in the preceding section. For ACS requirements, Single-Function Devices that are SR-IOV capable ~~or SIOV capable~~ must be handled as if they were Multi-Function Devices.

- ACS Source Validation: must not be implemented.
- ACS Translation Blocking: must not be implemented.
- ACS P2P Request Redirect: must be implemented by Functions that support peer-to-peer traffic with other Functions. This includes SR-IOV Virtual Functions (VFs). ~~SDIs that support peer-to-peer traffic with other Functions or SDIs in an SIOV Device must implement this in their associated PF. SIOV Devices must ensure that peer-to-peer traffic initiated by SDIs follow the policy set in their associated PF.~~

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

ACS P2P Request Redirect is subject to interaction with the ACS P2P Egress Control and ACS Direct Translated P2P mechanisms (if implemented). Refer to § Section 6.12.3 for more information.

When ACS P2P Request Redirect is enabled in a Multi-Function Device that is not an RCiEP, peer-to-peer Requests (between Functions of the device) must be redirected Upstream towards the RC.

It is permitted but not required to implement ACS P2P Request Redirect in an RCiEP. When ACS P2P Request Redirect is enabled in an RCiEP, peer-to-peer Requests, defined as all Requests that do not target system memory, must be sent to implementation specific logic within the Root Complex that determines whether the Request is directed towards its original target, or blocked as an ACS Violation error. The algorithms and specific controls for making this determination are not architected by this specification.

Completions are never affected by ACS P2P Request Redirect.

- ACS P2P Completion Redirect: must be implemented by Functions that implement ACS P2P Request Redirect. The intent of ACS P2P Completion Redirect is to avoid ordering rule violations between Completions and Requests when Requests are redirected. Refer to § Section 6.12.6 for more information.

ACS P2P Completion Redirect does not interact with ACS controls that govern Requests.

When ACS P2P Completion Redirect is enabled in a Multi-Function Device that is not an RCiEP, peer-to-peer Completions that do not have the Relaxed Ordering bit set must be redirected Upstream towards the RC. Otherwise, peer-to-peer Completions must ~~not~~ be ~~routed normally~~
~~redirected~~.

Errata: Base 6.3
B836 $\triangleleft\triangleright$

Requests are never affected by ACS P2P Completion Redirect.

- ACS Upstream Forwarding: must not be implemented.
- ACS P2P Egress Control: implementation is optional; is based on Function Numbers or Function Group Numbers ; controls peer-to-peer Requests between the different Functions within ~~a Multi-Function Device . SDIs within an SIOV Device do not have ACS P2P Egress Control of their own and follow the multi-Function or SR-IOV capable device. They follow the policy set in their associated PF.~~

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

ACS P2P Egress Control is subject to interaction with the ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms (if implemented). Refer to § Section 6.12.3 for more information.

Each Function within a Multi-Function Device that supports ACS P2P Egress Control can be selectively enabled to block peer-to-peer communication with other Functions or Function Groups¹⁴⁶ within the device. This is configured on a per Function basis.

With ACS P2P Egress ~~↑↓Control in multi-Function or SR-IOV capable devices,~~
~~↑↓Control,~~ controls in the ~~↑↓"sending"~~ ~~↑↓'sending'~~ Function determine if the Request is blocked, and if so, the ~~↑↓"sending"~~ ~~↑↓'sending'~~ Function handles the ACS Violation error per § Section 6.12.5. ~~↑↑For an SDI within an SIOV Device , controls in the associated PF determine if the Request is blocked, and if so, the associated PF handles the ACS Violation error per § Section 6.12.5.~~

ECN: Base 6.3
SIOV△↔

When ACS Function Groups are enabled in an ARI Device (ACS Function Groups Enable is Set), ACS P2P Egress Controls are enforced on a per Function Group basis instead of a per Function basis. See § Section 6.13.

Completions are never affected by ACS P2P Egress Control.

- ACS Direct Translated P2P: must be implemented if the ~~↑↓Multi-Function Device~~ Function supports Address Translation Services (ATS) and also peer-to-peer traffic with other Functions. ~~↑↑SDIs that support ATS and peer-to-peer traffic with other Functions or SDIs in an SIOV Device must implement this in their associated PF. SIOV Devices must ensure that peer-to-peer traffic initiated by SDIs follow the policy set in their associated PF.~~

ECN: Base 6.3
SIOV△↔

When ACS Direct Translated P2P is enabled in a Multi-Function Device, peer-to-peer Memory Requests whose Address Type (AT) field indicates a Translated address must be routed ~~↑↓normally ("directly")~~ ~~↑↑without redirection~~ to the peer Function, regardless of ACS P2P Request Redirect and ACS P2P Egress Control settings. All other peer-to-peer Requests must still be subject to ACS P2P Request Redirect and ACS P2P Egress Control settings.

ECN: Base 6.3
SIOV△↔

Completions are never affected by ACS Direct Translated P2P.

6.12.1.3 Functions in Single-Function Devices §

This section applies to Single-Function Device Functions, with the exception of Downstream Port Functions and SR-IOV capable ~~↑↑or SIOV capable~~ Functions, which are covered in a preceding section. For ACS requirements, Single-Function Devices that are SR-IOV capable ~~↑↑or SIOV capable~~ must be handled as if they were Multi-Function Devices.

ECN: Base 6.3
SIOV△↔

No ACS capabilities are applicable, and the Function must not implement an ACS Extended Capability structure.

6.12.2 Interoperability §

The following rules govern interoperability between ACS and non-ACS components:

- When ACS P2P Request Redirect and ACS P2P Completion Redirect are not being used, ACS and non-ACS components may be intermixed within a topology and will interoperate fully. ACS can be enabled in a subset of the ACS components without impacting interoperability.
- When ACS P2P Request Redirect, ACS P2P Completion Redirect, or both are being used, certain components in the PCI Express hierarchy must support ACS Upstream Forwarding (of Upstream redirected Requests). Specifically:

¹⁴⁶ ACS Function Groups capability is optional for ARI Devices that implement ACS P2P Egress Controls.

The associated Root Port¹⁴⁷ must support ACS Upstream Forwarding. Otherwise, how the Root Port handles Upstream redirected Request or Completion TLPs is undefined. The RC must also implement Redirected Request Validation.

Between each ACS component where P2P TLP redirection is enabled and its associated Root Port, any intermediate Switches must support ACS Upstream Forwarding. Otherwise, how such Switches handle Upstream redirected TLPs is undefined.

6.12.3 ACS Peer-to-Peer Control Interactions §

With each peer-to-peer Request, multiple ACS control mechanisms may interact to determine whether the Request is routed directly towards its peer-to-peer target, blocked immediately as an ACS Violation, or redirected Upstream towards the RC for access validation. Peer-to-peer Completion redirection is determined exclusively by the ACS P2P Completion Redirect mechanism.

If ACS Direct Translated P2P is enabled in a Port/Function, peer-to-peer Memory Requests whose Address Type (AT) field indicates a Translated address must be routed ~~normally ("directly")~~ to the peer ~~Port/Function~~, Port/Function without redirection, regardless of ACS P2P Request Redirect and ACS P2P Egress Control settings. Otherwise such Requests, and unconditionally all other peer-to-peer Requests, must be subject to ACS P2P Request Redirect and ACS P2P Egress Control settings. Specifically, the applicable Egress Control Vector bit, along with the ACS P2P Egress Control Enable bit (E) and the ACS P2P Request Redirect Enable bit (R), determine how the Request is handled. It must be noted that atomicity of accesses cannot be guaranteed if ACS peer-to-peer Request Redirect targets a legacy device location that can be the target of a locked access. Refer to § Section 7.7.11 for descriptions of these control bits. § Table 6-11 specifies the interactions.

Errata: Base 6.3
B836△◀

Table 6-11 ACS P2P Request Redirect and ACS P2P Egress Control Interactions §

| Control Bit E (b) | Control Bit R (b) | Egress Control Vector Bit for the Associated Egress Switch Port, Root Port, Function, or Function Group | Required Handling for Peer-to-Peer Requests |
|-------------------|-------------------|---|---|
| 0 | 0 | X - Don't care | Route directly to peer-to-peer target |
| 0 | 1 | X - Don't Care | Redirect Upstream |
| 1 | 0 | 1 | Handle as an ACS Violation |
| 1 | 0 | 0 | Route directly to peer-to-peer target |
| 1 | 1 | 1 | Redirect Upstream |
| 1 | 1 | 0 | Route directly to peer-to-peer target |

147. Not applicable for ACS Redirect between Functions of a multi-Function Root Complex Integrated Endpoint.

IMPLEMENTATION NOTE: ACCESS CONTROL SERVICES IN SYSTEMS THAT SUPPORT DIRECT ASSIGNMENT OF FUNCTIONS §

General-purpose VIs typically have separate address spaces for each SI and for the VI itself. If such a VI also supports direct assignment of a Function to an SI, Untranslated Memory Request transactions issued by directly assigned Functions are under the complete control of software operating within the associated SI and typically reference the address space associated with that SI. In contrast, Memory Request transactions issued by the Host (MMIO requests) and by Functions that are not directly assigned are under the control of the VI and typically reference one or more system address spaces (e.g., the PCIe physical address space, the address space associated with the VI, or the address space associated with some designated SI). General-purpose VIs are not expected to establish a dependency between these various address spaces. Consequently, these address spaces may freely overlap which could lead to unintended routing of TLPs by Switches. For example, Upstream Memory Request TLPs originated by a directly assigned Function and intended for main memory could instead be routed to a Downstream Port if the address in the Request falls within the MMIO address region associated with that Downstream Port. Such unintended routing poses a threat to SI and/or VI stability and integrity.

To guard against this concern, vendors are strongly recommended to implement ACS in platforms that support general purpose VIs with direct assignment of Functions. Such support should include:

- In Switches or Root Complexes located below the TA, the level of ACS support should follow the guidelines established in this document for Downstream Switch Ports that implement an ACS Extended Capability structure.

Note: Components located above the TA only see Translated Memory Requests, consequently this concern does not apply to those components.

- In ↑↓SR-IOV devices ↑↓SR-IOV Devices that are capable of peer-to-peer transactions, ACS support is required.
- In Multi-Function Devices that are capable of peer-to-peer transactions, vendors are strongly recommended to implement ACS with ACS P2P Egress Control.

Additionally, platform vendors should test for the presence of ACS and enable it in Root Complexes and Switches on the path from the TA to a Function prior to directly assigning that Function. If the Function is peer-to-peer capable, ACS should be enabled in the Function as well.

6.12.4 ACS Enhanced Capability §

ACS Enhanced Capability is an additional set of ACS control mechanisms to improve the level of isolation and protection provided by ACS. ACS Enhanced Capability defines the following additional access control mechanisms:

- ACS I/O Request Blocking
- ACS DSP Memory Target Access
- ACS USP Memory Target Access
- ACS Unclaimed Request Redirect

Through these mechanisms, ACS Enhanced Capability provides protection and consistent handling of Requests directed toward regions not covered by the original ACS mechanisms.

IMPLEMENTATION NOTE: ACS REDIRECT AND GUEST PHYSICAL ADDRESSES (GPAS) §

ACS redirect mechanisms were originally architected to enable fine-grained access control for P2P Memory Requests, by redirecting selected Requests Upstream to the RC, where validation logic determines whether to allow or deny access. However, ACS redirect mechanisms can also ensure that Functions under the direct control of VMs have their DMA Requests routed correctly to the Translation Agent in the host, which then translates their guest physical addresses (GPAs) into host physical addresses (HPAs).

GPA ranges used for Memory Space vs. DMA are not guaranteed to coincide with HPA ranges, which the PCIe fabric uses for Memory Request routing and access control. If any GPAs used for DMA fall within the HPA ranges used for Memory Space, legitimate or malicious packet misrouting can result.

ACS redirect mechanisms can ensure that Upstream Memory Requests with GPAs intended for DMA never get routed to HPA Memory ranges. ACS P2P Request Redirect handles this for (1) peer accesses between Functions within a Multi-Function Device and (2) peer accesses between Downstream Ports within a Switch or RC. ACS P2P Egress Control with redirect handles this in a more fine-grained manner for the same two cases.

Redirect mechanisms introduced with ACS Enhanced Capability handle this for additional cases. ACS DSP Memory Target Access with redirect handles this for Downstream Port Memory Resource ranges. ACS USP Memory Target Access with redirect handles this for Switch Upstream Port Memory Resource ranges. In Switches, ACS Unclaimed Request Redirect handles this for any areas within Upstream Port Memory apertures that are not handled by the other ACS redirect mechanisms.

Together these ACS redirect mechanisms can ensure that Upstream Memory Requests with GPAs intended for DMA are always routed or redirected to the Translation Agent in the host, and those with GPAs intended for P2P are still routed as originally architected.

6.12.5 ACS Violation Error Handling §

ACS Violations may occur due to either hardware or software defects/failures. To assist in fault isolation and root cause analysis, it is recommended that AER be implemented in ACS components. AER prefix/header logging and the Prefix Log/Header Log registers may be used to determine the prefix/header of the offending Request. The ACS Violation Status, Mask, and Severity bits provide positive identification of the error and increased control over error logging and signaling.

When an ACS Violation is detected, the ~~detecting~~ ACS component ~~that~~ operates as the Completer.¹⁴⁸ ~~and~~ must do the following:

Errata: Base 6.3
B818△◀▷

- For Non-Posted ~~and UIO~~ Requests, the Completer must generate a Completion with a Completer Abort (CA) Completion ~~Status~~ ~~(see § Section 2.2.9.1).~~

Errata: Base 6.3
B818△◀▷

- ~~This~~ Completer must log and signal the ACS Violation as indicated in § Figure 6-2. Note the following:

Errata: Base 6.3
B818△◀▷

- Even though the Completer uses a CA Completion Status when it sends a Completion, the Completer must log an ACS Violation error instead of a Completer Abort error.

148. In all cases but one, the ACS component that detects the ACS Violation also operates as the Completer. The exception case is when Root Complex Redirected Request Validation logic disallows a redirected Request. If the redirected Request came through a Root Port, that Root Port must operate as the Completer. If the redirected Request came from a Root Complex Integrated Endpoint, the associated Root Complex Event Collector must operate as the Completer.

- If the severity of the ACS Violation is non-fatal and the Completer sends a Completion with CA Completion Status, this case must be handled as an Advisory Non-Fatal Error as described in § Section 6.2.3.2.4.1.

~~↓↓The↓~~

- ~~↑↑This↑~~ Completer¹⁴⁹ must set the ~~↓↓Signaled Target Abort↓~~ ~~↑↑Signaled Target Abort↑~~ bit in either its Status register or Secondary Status register as appropriate.

Errata: Base 6.3
B818△◀

6.12.6 ACS Redirection Impacts on Ordering Rules §

When ACS P2P Request Redirect is enabled, some or all peer-to-peer Requests are redirected, which can cause ordering rule violations in some cases. This section explores those cases, plus a similar case that occurs with RCs that implement “Request Retargeting” as an alternative mechanism for enforcing peer-to-peer access control.

6.12.6.1 Completions Passing Posted Requests §

When a peer-to-peer Posted Request is redirected, a subsequent peer-to-peer non-RO¹⁵⁰ Completion that is routed directly can effectively pass the redirected Posted Request, violating the ordering rule that non-RO Completions must not pass Posted Requests. Refer to § Section 2.4.1 for more information.

ACS P2P Completion Redirect can be used to avoid violating this ordering rule. When ACS P2P Completion Redirect is enabled, all peer-to-peer non-RO Completions will be redirected, thus taking the same path as redirected peer-to-peer Posted Requests. Enabling ACS P2P Completion Redirect when some or all peer-to-peer Requests are routed directly will not cause any ordering rule violations, since it is permitted for a given Completion to be passed by any TLP other than another Completion with the same Transaction ID.

As an alternative mechanism to ACS P2P Request Redirect for enforcing peer-to-peer access control, some RCs implement “Request Retargeting”, where the RC supports special address ranges for “peer-to-peer” traffic, and the RC will retarget validated Upstream Requests to peer devices. Upon receiving an Upstream Request targeting a special address range, the RC validates the Request, translates the address to target the appropriate peer device, and sends the Request back Downstream. With retargeted Requests that are Non-posted, if the RC does not modify the Requester ID, the resulting Completions will travel “directly” peer-to-peer back to the original Requester, creating the possibility of non-RO Completions effectively passing retargeted Posted Requests, violating the same ordering rule as when ACS P2P Request Redirect is being used. ACS P2P Completion Redirect can be used to avoid violating this ordering rule here as well.

If ACS P2P Request Redirect and RC P2P Request Retargeting are not being used, there is no envisioned benefit to enabling ACS P2P Completion Redirect, and it is recommended not to do so because of potential performance impacts.

149. Similarly, if the Request was ~~↓↓Non-Posted,↓~~ ~~↑↑a Non-Posted or UIO Request,↑~~ when the Requester receives the resulting Completion with CA Completion Status, the Requester must set the ~~↓↓Received Target Abort↓~~ ~~↑↑Received Target Abort↑~~ bit in either its Status register or Secondary Status register as appropriate. Note that for the case of a Multi-Function Device incurring an ACS Violation error with a peer-to-peer Request between its Functions, the same Function might serve both as Requester and Completer.

150. In this section, “non-RO” is an abbreviation characterizing TLPs whose Relaxed Ordering Attribute field is not set.

IMPLEMENTATION NOTE: PERFORMANCE IMPACTS WITH ACS P2P COMPLETION REDIRECT §

While the use of ACS P2P Completion Redirect can avoid ordering violations with Completions passing Posted Requests, it also may impact performance. Specifically, all redirected Completions will have to travel up to the RC from the point of redirection and back, introducing extra latency and possibly increasing Link and RC congestion.

Since peer-to-peer Completions with the Relaxed Ordering bit set are never redirected (thus avoiding performance impacts), it is strongly recommended that Requesters be implemented to maximize the proper use of Relaxed Ordering, and that software enable Requesters to utilize Relaxed Ordering by setting the Enable Relaxed Ordering bit in the Device Control Register.

If software enables ACS P2P Request Redirect, RC P2P Request Retargeting, or both, and software is certain that proper operation is not compromised by peer-to-peer non-RO Completions passing peer-to-peer¹⁵¹ Posted Requests, it is recommended that software leave ACS P2P Completion Redirect disabled as a way to avoid its performance impacts.

6.12.6.2 Requests Passing Posted Requests §

When some peer-to-peer Requests are redirected but other peer-to-peer Requests are routed directly, the possibility exists of violating the ordering rules where ~~↓↑Non-posted↓~~ ↑↑Non-Posted↓ Requests or non-RO Posted Requests must not pass Posted Requests. Refer to § Section 2.4.1 for more information.

These ordering rule violation possibilities exist only when ACS P2P Request Redirect and ACS Direct Translated P2P are both enabled. Software should not enable both these mechanisms unless it is certain either that such ordering rule violations cannot occur, or that proper operation will not be compromised if such ordering rule violations do occur.

¹⁵¹. These include true peer-to-peer Requests that are redirected by the ACS P2P Request Redirect mechanism, as well as “logically peer-to-peer” Requests routed to the Root Complex that the Root Complex then retargets to the peer device.

IMPLEMENTATION NOTE: ENSURING PROPER OPERATION WITH ACS DIRECT TRANSLATED P2P §

The intent of ACS Direct Translated P2P is to optimize performance in environments where Address Translation Services (ATS) are being used with peer-to-peer communication whose access control is enforced by the RC.

Permitting peer-to-peer Requests with Translated addresses to be routed directly avoids possible performance impacts associated with redirection, which introduces extra latency and may increase Link and RC congestion.

For the usage model where peer-to-peer Requests with Translated addresses are permitted, but those with Untranslated addresses are to be blocked as ACS Violations, it is recommended that software enable ACS Direct Translated P2P and ACS P2P Request Redirect, and configure the Redirected Request Validation logic in the RC to block the redirected Requests with Untranslated addresses. This configuration has no ordering rule violations associated with Requests passing Posted Requests.

For the usage model where some Requesters use Translated addresses exclusively with peer-to-peer Requests and some Requesters use Untranslated addresses exclusively with peer-to-peer Requests, and the two classes of Requesters do not communicate peer-to-peer with each other, proper operation is unlikely to be compromised by redirected peer-to-peer Requests (with Untranslated addresses) being passed by direct peer-to-peer Requests (with Translated addresses). It is recommended that software not enable ACS Direct Translated P2P unless software is certain that proper operation is not compromised by the resulting ordering rule violations.

For the usage model where a single Requester uses both Translated and Untranslated addresses with peer-to-peer Requests, again it is recommended that software not enable ACS Direct Translated P2P unless software is certain that proper operation is not compromised by the resulting ordering rule violations. This requires a detailed analysis of the peer-to-peer communications models being used, and is beyond the scope of this specification.

6.13 Alternative Routing-ID Interpretation (ARI) §

Routing IDs, Requester IDs, and Completer IDs are 16-bit identifiers traditionally composed of three fields: an 8-bit Bus Number, a 5-bit Device Number, and a 3-bit Function Number. With ARI, the 16-bit field is interpreted as two fields instead of three: an 8-bit Bus Number and an 8-bit Function Number - the Device Number field is eliminated. This new interpretation enables an ARI Device to support up to 256 Functions [0..255] instead of 8 Functions [0..7].

ARI is controlled by a new set of optional capability and control register bits. These provide:

- Software the ability to detect whether a component supports ARI.
- Software the ability to configure an ARI Downstream Port so the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.
- Software the ability to configure an ARI Device to assign each Function to a Function Group. Controls based on Function Groups may be preferable when finer granularity controls based on individual Functions are not required.
 - If Multi-Function VC arbitration is supported and enabled, arbitration can optionally be based on Function Groups instead of individual Functions.

- If ACS P2P Egress Controls are supported and enabled, access control can optionally be based on Function Groups instead of individual Functions.

The following illustrates an example flow for enabling these capabilities and provides additional details on their usage:

1. Software enumerates the PCI Express hierarchy and determines whether ARI is supported.
 - a. For an ARI Downstream Port, the capability is communicated through the Device Capabilities 2 register.
 - b. For an ARI Device, the capability is communicated through the ARI Extended Capability structure.
 2. Software enables ARI functionality in each component.
 - a. In an ARI Downstream Port immediately above an ARI Device, software sets the ARI Forwarding Enable bit in the Device Control 2 register. Setting this bit ensures the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.
 - b. In an ARI Device, Extended Functions must respond if addressed with a Type 0 Configuration Request. It is necessary for ARI-aware software to enable ARI Forwarding in the Downstream Port immediately above the ARI Device, in order for ARI-aware software to discover and configure the Extended Functions.
 - c. If an ARI Device implements a Multi-Function VC Extended Capability structure with Function arbitration, and also implements MFVC Function Groups, ARI-aware software categorizes Functions into Function Groups.
 - i. Each Function is assigned to a Function Group represented by a **Function Group Number**.
 - ii. A maximum of 8 Function Groups can be configured.
 - iii. Within the Multi-Function VC Arbitration Table, a Function Group Number is used in place of a Function Number in each arbitration slot.
 1. Arbitration occurs on a Function Group basis instead of an individual Function basis.
 2. All other aspects of Multi-Function VC arbitration remain unchanged. See § Section 7.9.2.10 for additional details.
 - iv. Function arbitration within each Function Group is implementation specific.
 - d. If an ARI Device supports ACS P2P Egress Control, access control can be optionally implemented on a Function Group basis.
 - e. To improve the enumeration performance and create a more deterministic solution, software can enumerate Functions through a linked list of Function Numbers. The next linked list element is communicated through each Function's ARI Capability Register.
 - i. Function 0 acts as the head of a linked list of Function Numbers. Software detects a non-Zero Next Function Number field within the ARI Capability Register as the next Function within the linked list. Software issues a configuration probe using the Bus Number captured by the Device and the Function Number derived from the ARI Capability Register to locate the next associated Function's configuration space.
 - ii. Function Numbers may be sparse and non-sequential in their consumption by an ARI Device.

With an ARI Device, the Phantom Functions Supported field within each Function's Device Capabilities register (see § Section 7.5.3.3 , § Table 7-20) must be set to 00b to indicate that Phantom Functions are not supported. The

Errata: Base 6.3

[Requester Enable](#) bit can still be used to enable each Function to support higher numbers of outstanding [non-UIO Requests](#). See § [Section 2.2.6.2](#).

§ Figure 6-13 shows an example system topology with two ARI Devices, one below a Root Port and one below a Switch. For access to Extended Functions in ARI Device X, Root Port A must support ARI Forwarding and have it enabled by software. For access to Extended Functions in ARI Device Y, Switch Downstream Port D must support ARI Forwarding and have it enabled by software. With this configuration, it is recommended that software not enable ARI Forwarding in Root Port B or Switch Downstream Port C.

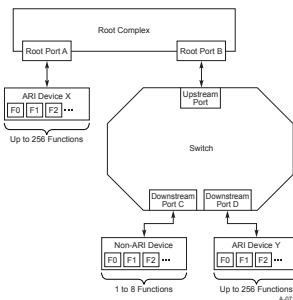


Figure 6-13 Example System Topology with ARI Devices §

IMPLEMENTATION NOTE: ARI FORWARDING ENABLE BEING SET INAPPROPRIATELY §

It is strongly recommended that software in general Set the ARI Forwarding Enable bit in a Downstream Port only if software is certain that the device immediately below the Downstream Port is an ARI Device. If the bit is Set when a non-ARI Device is present, the non-ARI Device can respond to Configuration Space accesses under what it interprets as being different Device Numbers, and its Functions can be aliased under multiple Device Numbers, generally leading to undesired behavior.

Following a hot-plug event below a Downstream Port, it is strongly recommended that software Clear the ARI Forwarding Enable bit in the Downstream Port until software determines that a newly added component is in fact an ARI Device.

IMPLEMENTATION NOTE: ARI FORWARDING ENABLE SETTING AT FIRMWARE/ OPERATING SYSTEM CONTROL HANDOFF §

It is strongly recommended that firmware not have the ARI Forwarding Enable bit Set in a Downstream Port upon control handoff to an operating system unless firmware knows that the operating system is ARI-aware. With this bit Set, a non-ARI-aware operating system might be able to discover and enumerate Extended Functions in an ARI Device below the Downstream Port, but such an operating system would generally not be able to manage Extended Functions successfully, since it would interpret there being multiple Devices below the Downstream Port instead of a single ARI Device. As one example of many envisioned problems, the interrupt binding for INTx virtual wires would not be consistent with what the non-ARI-aware operating system would expect.

6.14 Multicast Operations §

The Multicast Capability structure defines a Multicast address range, the segmentation of that range into a number, N, of equal sized Multicast Windows, and the association of each Multicast Window with a Multicast Group, MCG. Each Function that supports Multicast within a component implements a Multicast Capability structure that provides routing directions and permission checking for each MCG for TLPs passing through or to the Function. The Multicast Group is a field of up to 6 bits in width embedded in the address beginning at the MC_Index_Position , as defined in § Section 7.9.11.4 .

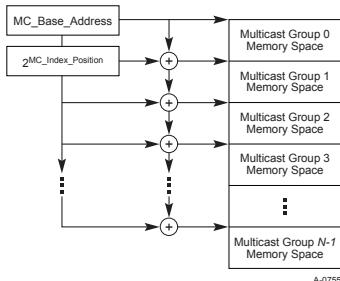


Figure 6-14 Segmentation of the Multicast Address Range §

6.14.1 Multicast TLP Processing §

A Multicast Hit occurs if all of the following are true:

- MC_Enable is Set
 - TLP is a non-UIO Memory Write Request or an Address Routed Message, both of which are non-UIO Posted Requests
 - Address $TLP \geq MC_Base_Address$
 - Address $TLP < (MC_Base_Address + (2^{\underline{MC_Index_Position}} * (MC_Num_Group + 1)))$

In this step, each Switch Ingress Port and other components use values of MC_Enable , MC_Base_Address , MC_Index_Position , and MC_Num_Group from any one of their Functions. Software is required to configure all Functions of a Switch and all Functions of a Multi-Function Upstream Port to have the same values in each of these fields and results are indeterminate if this is not the case.

If the address in a Non-Posted Memory Request hits in a Multicast Window, no Multicast Hit occurs and the TLP is processed normally per the base specification - i.e., as a unicast.

If a Multicast Hit occurs, the only ACS access control that can still apply is ACS Source Validation. In particular, neither ACS P2P Request Redirect nor ACS P2P Egress Control affects operations during a Multicast Hit. See § Section 6.12 .

If a Multicast Hit occurs, normal address routing rules do not apply. Instead, the TLP is processed as follows:

The Multicast Group is extracted from the address in the TLP using any Function's values for MC_Base_Address and MC_Index_Position. Specifically:

$$MCG = ((Address_{TIP} - MC\ Base\ Address) \gg MC\ Index\ Position) \& 3Fh$$

In this process, the component may use any Function's values for MC_Base_Address and MC_Index_Position. Which Function's values are used is device-specific.

Components next check the MC_Block_All and the MC_Block_Untranslated bits corresponding to the extracted MCG. Switches and Root Ports check Multicast TLPs in their Ingress Ports using the MC_Block_All and MC_Block_Untranslated registers associated with the Ingress Port. Endpoint Functions check Multicast TLPs they are preparing to send, using their MC_Block_All and MC_Block_Untranslated registers. If the MC_Block_All bit corresponding to the extracted MCG is set, the TLP is handled as an MC Blocked TLP. If the MC_Block_Untranslated bit corresponding to the extracted MCG is set and the TLP contains an Untranslated Address, the TLP is also handled as an MC Blocked TLP.

IMPLEMENTATION NOTE: MC_BLOCK_UNTRANSLATED AND PIO WRITES §

Programmed I/O (PIO) Writes to Memory Space generally have Untranslated addresses since there is no architected mechanism for software to control the Address Type (AT) field for PIO Requests. Thus, if it's necessary for a given Switch to Multicast any PIO Writes, software should ensure that the appropriate MC_Block_Untranslated bits in the Upstream Port of that Switch are Clear. Otherwise, the Switch Upstream Port may block PIO Writes that legitimately target Multicast Windows. Since it may be necessary for software to clear MC_Block_Untranslated bits in a Switch Upstream Port for the sake of PIO Writes, the following are strongly recommended for a Root Complex capable of Address translation:

- All Integrated Endpoints each implement a Multicast Capability structure to provide access control for sending Untranslated Multicast TLPs.
- All peer-to-peer capable Root Ports each implement a Multicast Capability structure to provide access control for Untranslated Multicast TLPs that are forwarded peer-to-peer.

For similar reasons, with Multicast-capable Switch components where the Upstream Port is a Function in a Multi-Function Device , it is strongly recommended that any Endpoints in that Multi-Function Device each implement a Multicast Capability structure.

IMPLEMENTATION NOTE: MULTICAST WINDOW SIZE §

Each ultimate Receiver of a Multicast TLP may have a different Multicast Window size requirement. At one extreme, a Multicast Window may be required to cover a range of memory implemented within the device. At the other, it may only need to cover a particular offset at which a FIFO register is located. The MC_Window_Size_Requested field within the Multicast Capability register is used by an Endpoint to advertise the size of Multicast Window that it requires.

Unless available address space is limited, resource allocation software may be able to treat each request as a minimum and set the Multicast Window size via MC_Index_Position to accommodate the largest request. In some cases, a request for a larger window size can be satisfied by configuring a smaller window size and assigning the same membership to multiple contiguous MCGs.

IMPLEMENTATION NOTE: MULTICAST, ATS, AND REDIRECTION §

The ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms provide a means where P2P Requests with Untranslated Addresses can be redirected to the Root Complex (RC) for access control checking, whereas P2P Requests with Translated Addresses can be routed “directly” to their P2P targets for improved performance. See § [Section 6.12](#). No corresponding redirection mechanism exists for Multicast TLPs.

To achieve similar functionality, an RC might be configured to provide one or more target Memory Space ranges that are not in the Multicast address range, but the RC maps to “protected” Multicast Windows. Multicast TLP senders either with or without ATS capability then target these RC Memory Space ranges in order to access the protected Multicast Windows indirectly. When either type of sender targets these ranges with Memory Writes, each TLP that satisfies the access control checks will be reflected back down by the RC with a Translated Address targeting a protected Multicast Window.¹⁵² ATS-capable senders can request and cache Translated Addresses using the RC Memory Space range, and then later use those Translated Addresses for Memory Writes that target protected Multicast Windows directly and can be Multicast without a taking a trip through the RC.

For hardware enforcement that only Translated Addresses can be used to target the protected Multicast Windows directly, software Sets appropriate MCG bits in the [MC_Block_Untranslated](#) register in all applicable Functions throughout the platform. Each MCG whose bit is Set will cause its associated Multicast Window to be protected from direct access using Untranslated Addresses.

If the TLP is not blocked in a Switch or Root Complex it is forwarded out all of the Ports, except its Ingress Port, whose MC_Receive bit corresponding to the extracted MCG is set. In an Endpoint, it is consumed by all Functions whose MC_Receive bit corresponding to the extracted MCG is set. If no Ports forward the TLP or no Functions consume it, the TLP is silently dropped.

To prevent loops, it is prohibited for a Root Port or a Switch Port to forward a TLP back out its Ingress Port, even if so specified by the MC_Receive register associated with the Port. An exception is the case described in the preceding Implementation Note, where an RC reflects a unicast TLP that came in on an Ingress Root Port to a Multicast Window. In that case, when specified by the MC_Receive register associated with that Ingress Root Port, the RC is required to send the reflected TLP out the same Root Port that it originally came in.

A Multicast Hit suspends normal address routing, including default Upstream routing in Switches. When a Multicast Hit occurs, the TLP will be forwarded out only those Egress Ports whose MC_Receive bit associated with the MCG extracted from the address in the TLP is set. If the address in the TLP does not decode to any Downstream Port using normal address decode, the TLP will be copied to the Upstream Port only if so specified by the Upstream Port’s MC_Receive register.

6.14.2 Multicast Ordering §

No new ordering rules are defined for processing Multicast TLPs. All Multicast TLPs are Posted Requests and follow Posted Request ordering rules. Multicast TLPs are ordered per normal ordering rules relative to other TLPs in a component’s ingress stream through the point of replication. Once copied into an egress stream, a Multicast TLP follows the same ordering as other Posted Requests in the stream.

¹⁵². If the original sender belongs to the MCG associated with this Window, the original sender will also receive a copy of the reflected TLP.

6.14.3 Multicast Capability Structure Field Updates §

Some fields of the Multicast Capability structure may be changed at any time. Others cannot be changed with predictable results unless the MC_Enable bit is Clear in every Function of the component. The latter group includes MC_Base_Address and MC_Index_Position .

Fields which software may change at any time include MC_Enable , MC_Num_Group , MC_Receive, MC_Block_All , and MC_Block_Untranslated . Updates to these fields must themselves be ordered. Consider, for example, TLPs A and B arriving in that order at the same Ingress Port and in the same TC. If A uses value X for one of these fields, then B must use the same value or a newer value.

For Multi-Function Upstream Switch Ports Multicast TLPs received by one Switch or transmitted by one Endpoint Function are presented to the other parallel Endpoint Functions and the Downstream Switch Ports of the other parallel Switches (Functions are considered to be parallel if they are in the same Device. A single Multicast TLP is forwarded Upstream when any of the Upstream Switch Functions has the appropriate MC_Receive bit Set.

6.14.4 MC Blocked TLP Processing §

When a TLP is blocked by the MC_Block_All or the MC_Block_Untranslated mechanisms, the TLP is dropped. The Function blocking the TLP serves as the Completer. The Completer must log and signal this MC Blocked TLP error as indicated in § Figure 6-2 . In addition, the Completer must set the Signaled Target Abort bit in either its Status register or Secondary Status register as appropriate. To assist in fault isolation and root cause analysis, it is strongly recommended that AER be implemented in Functions with Multicast capability.

In Root Complexes and Switches, if the error occurs with a TLP received by an Ingress Port, the error is reported by that Ingress Port. If the error occurs in an Endpoint Function preparing to send the TLP, the error is reported by that Endpoint Function.

6.14.5 MC_Overlay Mechanism §

The MC_Overlay mechanism is provided to allow a single BAR in an Endpoint that doesn't contain a Multicast Capability structure to be used for both Multicast and unicast TLP reception. Software can configure the MC_Overlay mechanism to affect this by setting the MC_Overlay_BAR in a Downstream Port so that the Multicast address range, or a portion of it, is remapped (overlaid) onto the Memory Space range accepted by the Endpoint's BAR. At the Upstream Port of a Switch, the mechanism can be used to overlay a portion of the Multicast address range onto a Memory Space range associated with host memory.

A Downstream Port's MC_Overlay mechanism applies to TLPs exiting that Port. An Upstream Port's MC_Overlay mechanism applies to TLPs exiting the Switch heading Upstream. A Port's MC_Overlay mechanism does not apply to TLPs received by the Port, to TLPs targeting memory space within the Port, or to TLPs routed Peer-to-Peer between Functions in a Multi-Function Upstream Port.

When enabled, the overlay operation specifies that bits in the address in the Multicast TLP, whose bit numbers are equal to or higher than the MC_Overlay_Size field, be replaced by the corresponding bits in the MC_Overlay_BAR . In other words:

```
If ( MC_Overlay_Size < 6 )
    Then Egress_TLP_Addr = Ingress_TLP_Addr;
Else Egress_TLP_Addr = { MC_Overlay_BAR [63: MC_Overlay_Size ],
                        Ingress_TLP_Addr[ MC_Overlay_Size -1:0] };

```

Equation 6-1 MC_Overlay Transform rules §

If the TLP with modified address contains the optional ECRC, the unmodified ECRC will almost certainly indicate an error. The action to be taken if a TLP containing an ECRC is Multicast copied to an Egress Port that has MC_Overlay enabled depends upon whether or not optional support for ECRC regeneration is implemented. All of the contingent actions are outlined in § Table 6-12 . If MC_Overlay is not enabled, the TLP is forwarded unmodified. If MC_Overlay is enabled and the TLP has no ECRC, the modified TLP, with its address replaced as specified in the previous paragraph is forwarded. If the TLP has an ECRC but ECRC regeneration is not supported, then the modified TLP is forwarded with its ECRC dropped and the TD bit in the header cleared to indicate no ECRC attached. If the TLP has an ECRC and ECRC regeneration is supported, then an ECRC check is performed before the TLP is forwarded. If the ECRC check passes, the TLP is forwarded with regenerated ECRC. If the ECRC check fails, the TLP is forwarded with inverted regenerated ECRC.

Table 6-12 ECRC Rules for MC_Overlay §

| MC_Overlay Enabled | TLP has ECRC | ECRC Regeneration Supported | Action if ECRC Check Passes | Action if ECRC Check Fails |
|--------------------|--------------|-----------------------------|---|---|
| No | x | x | Forward TLP unmodified | |
| Yes | No | x | Forward modified TLP | |
| Yes | Yes | No | Forward modified TLP with ECRC dropped and TD bit clear | |
| Yes | Yes | Yes | Forward modified TLP with regenerated ECRC | Forward modified TLP with inverted regenerated ECRC |

IMPLEMENTATION NOTE: MC_OVERLAY AND ECRC REGENERATION §

Switch and Root Complex Ports have the option to support ECRC regeneration. If ECRC regeneration is supported, then it is strongly recommended to do so robustly by minimizing the time between checking the ECRC of the original TLP and replacing it with an ECRC computed on the modified TLP. The TLP is unprotected during this time, leaving a data integrity hole if the pre-check and regeneration aren't accomplished in the same pipeline stage.

Stripping the ECRC from Multicast TLPs passing through a Port that has MC_Overlay enabled but doesn't support ECRC regeneration allows the receiving Endpoint to enable ECRC checking. In such a case, the Endpoint will enjoy the benefits of ECRC on non-Multicast TLPs without detecting ECRC on Multicast TLPs modified by the MC_Overlay mechanism.

When Multicast ECRC regeneration is supported, and an ECRC error is detected prior to TLP modification, then inverting the regenerated ECRC ensures that the ECRC error isn't masked by the regeneration process.

IMPLEMENTATION NOTE: MULTICAST TO ENDPOINTS THAT DON'T HAVE MULTICAST CAPABILITY §

An Endpoint Function that doesn't contain a Multicast Capability structure cannot distinguish Multicast TLPs from unicast TLPs. It is possible for a system designer to take advantage of this fact to employ such Endpoints as Multicast targets. The primary requirement for doing so is that the base and limit registers of the virtual PCI to PCI Bridge in the Switch Port above the device be configured to overlap at least part of the Multicast address range or that the MC_Overlay mechanism be employed. Extending this reasoning, it is even possible that a single Multicast target Function could be located on the PCI/PCI-X side of a PCI Express to PCI/PCI-X Bridge.

If an Endpoint without a Multicast Capability structure is being used as a Multicast target and the MC_Overlay mechanism isn't used, then it may be necessary to read from the Endpoint's Memory Space using the same addresses used for Multicast TLPs. Therefore, Memory Reads that hit in a Multicast Window aren't necessarily errors. Memory Reads that hit in a Multicast Window and that don't also hit in the aperture of an RCiEP or the Downstream Port of a Switch will be routed Upstream, per standard address routing rules, and be handled as a UR there.

IMPLEMENTATION NOTE: MULTICAST IN A ROOT COMPLEX §

A Root Complex with multiple Root Ports that supports Multicast may implement as many Multicast Capability structures as its implementation requires. If it implements more than one, software should ensure that certain fields, as specified in § Section 6.14.3 , are configured identically. To support Multicast to RCiEPs, the implementation needs to expose all TLPs identified as Multicast via the MC_Base_Address register to all potential Multicast target Endpoints integrated within it. Each such Integrated Endpoint then uses the MC_Receive register in its Multicast Capability structure to determine if it should receive the TLP.

IMPLEMENTATION NOTE: MULTICAST AND MULTI-FUNCTION DEVICES §

All Port Functions and Endpoint Functions that are potential Multicast targets need to implement a Multicast Capability structure so that each has its own MC_Receive vector. Within a single component, software should configure the MC_Enable , MC_Base_Address , MC_Index_Position , and MC_Num_Group fields of these Capability structures identically. That being the case, it is sufficient to implement address decoding logic on only one instance of the Multicast BAR in the component.

IMPLEMENTATION NOTE: CONGESTION AVOIDANCE §

The use of Multicast increases the output link utilization of Switches to a degree proportional to both the size of the Multicast groups used and the fraction of Multicast traffic to total traffic. This results in an increased risk of congestion and congestion spreading when Multicast is used.

To mitigate this risk, components that are intended to serve as Multicast targets should be designed to consume Multicast TLPs at wire speed. Components that are intended to serve as Multicast sources should consider adding a rate limiting mechanism.

In many applications, the application's Multicast data flow will have an inherent rate limit and can be accommodated without causing congestion. Others will require an explicit mechanism to limit the injection rate, selection of a Switch with buffers adequate to hold the requisite bursts of Multicast traffic without asserting flow control, or selection of Multicast target components capable of sinking the Multicast traffic at the required rate. It is the responsibility of the system designer to choose the appropriate mechanisms and components to serve the application.

IMPLEMENTATION NOTE: THE HOST AS A MULTICAST RECIPIENT §

For general-purpose systems, it is anticipated that the Multicast address range will usually not be configured to overlap with Memory Space that's directly mapped to host memory. If host memory is to be included as a Multicast recipient, the Root Complex may need to have some sort of I/O Memory Management Unit (IOMMU) that is capable of remapping portions of Multicast Windows to host memory, perhaps with page-level granularity. Alternatively, the MC_Overlay mechanism in the Upstream Port of a Switch can be used to overlay a portion of the Multicast address range onto host memory.

For embedded systems that lack an IOMMU, it may be feasible to configure Multicast Windows overlapping with Memory Space that's directly mapped to host memory, thus avoiding the need for an IOMMU. Specific details of this approach are beyond the scope of this specification.

6.15 Atomic Operations (AtomicOps) §

An Atomic Operation (AtomicOp) is a single PCI Express transaction that targets a location in Memory Space, reads the location's value, potentially writes a new value back to the location, and returns the original value. This “read-modify-write” sequence to the location is performed atomically. AtomicOps include the following:

- FetchAdd (Fetch and Add): Request contains a single operand, the “add” value
 - Read the value of the target location.
 - Add the “add” value to it using two's complement arithmetic ignoring any carry or overflow.
 - Write the sum back to the target location.
 - Return the original value of the target location.
- Swap (Unconditional Swap): Request contains a single operand, the “swap” value

- Read the value of the target location.
- Write the “swap” value back to the target location.
- Return the original value of the target location.
- CAS (Compare and Swap): Request contains two operands, a “compare” value and a “swap” value
 - Read the value of the target location.
 - Compare that value to the “compare” value.
 - If equal, write the “swap” value back to the target location.
 - Return the original value of the target location.

A given AtomicOp transaction has an associated operand size, and the same size is used for the target location accesses and the returned value. FetchAdd and Swap support operand sizes of 32 and 64 bits. CAS supports operand sizes of 32, 64, and 128 bits.

AtomicOp capabilities are optional normative. Endpoints and Root Ports are permitted to implement AtomicOp Requester capabilities. PCI Express Functions with Memory Space BARs as well as all Root Ports are permitted to implement AtomicOp Completer capabilities. Routing elements (Switches, as well as Root Complexes supporting peer-to-peer access between Root Ports) require AtomicOp routing capability in order to route AtomicOp Requests. AtomicOps are architected for device-to-host, device-to-device, and host-to-device transactions. In each case, the Requester, Completer, and all intermediate routing elements must support the associated AtomicOp capabilities.

AtomicOp capabilities are not supported on PCI Express to PCI/PCI-X Bridges. If need be, Locked Transactions can be used for devices below such Bridges. AtomicOps and Locked Transactions can operate concurrently on the same hierarchy.

Software discovers specific AtomicOp Completer capabilities via three bits in the Device Capabilities 2 register (see § Section 7.5.3.15). For increased interoperability, Root Ports are required to implement certain AtomicOp Completer capabilities in sets if at all (see § Section 6.15.3.1). Software discovers AtomicOp routing capability via the AtomicOp Routing Supported bit in the Device Capabilities 2 register. Software discovery of AtomicOp Requester capabilities is outside the scope of this specification, but software must set the AtomicOp Requester Enable bit in a Function’s Device Control 2 register before the Function can initiate AtomicOp Requests (see § Section 7.5.3.16).

With routing elements, software can Set an AtomicOp Egress Blocking bit (see § Section 7.5.3.16) on a Port-by-Port basis to avoid AtomicOp Requests being forwarded to components that shouldn’t receive them, and might, if operating in NFM, handle each as a Malformed TLP, which by default is a Fatal Error. Each blocked Request is handled as an AtomicOp Egress Blocked error, which by default is an Advisory Non-Fatal Error. For Root Ports, to ensure that AtomicOps are not transmitted on a given Link, software should Clear AtomicOp Requester Enable and Set AtomicOp Egress Blocking.

The AtomicOp Requester Enable bit, when Clear, prevents transmission of AtomicOps initiated by an entity. When AtomicOps are initiated by a programmable entity (e.g., CPU or accelerator Function), this specification does not architect a mechanism for reporting errors when the bit is Clear and software attempts to initiate an AtomicOp. For Root Ports, when AtomicOp Requester Enable is Clear and AtomicOp Egress Blocking is Set, it is outside the scope of this specification which error mechanism (or both) will be triggered if software attempts to initiate an AtomicOp.

AtomicOps are Memory Transactions, so existing standard mechanisms for managing Memory Space access (e.g., Bus Master Enable, Memory Space Enable, and Base Address registers) apply.

6.15.1 AtomicOp Use Models and Benefits §

AtomicOps enable advanced synchronization mechanisms that are particularly useful when there are multiple producers and/or multiple consumers that need to be synchronized in a non-blocking fashion. For example, multiple producers can safely enqueue to a common queue without any explicit locking.

AtomicOps also enable lock-free statistics counters, for example where a device can atomically increment a counter, and host software can atomically read and clear the counter.

Direct support for the three chosen AtomicOps over PCI Express enables easier migration of existing high-performance SMP applications to systems that use PCI Express as the interconnect to tightly-coupled accelerators, co-processors, or GP-GPUs. For example, a ported application that uses PCI Express-attached accelerators may be able to use the same synchronization algorithms and data structures as the earlier SMP application.

An AtomicOp to a given target generally incurs latency comparable to a Memory Read to the same target. Within a single hierarchy, multiple AtomicOps can be “in flight” concurrently. AtomicOps generally create negligible disruption to other PCI Express traffic.

Compared to Locked Transactions, AtomicOps provide lower latency, higher scalability, advanced synchronization algorithms, and dramatically less impact to other PCI Express traffic.

6.15.2 AtomicOp Transaction Protocol Summary §

Detailed protocol rules and requirements for AtomicOps are distributed throughout the rest of this specification, but here is a brief summary plus some unique requirements.

- AtomicOps are Non-Posted Memory Transactions, supporting 32- and 64-bit address formats.
- FetchAdd, Swap, and CAS each use a distinct type code.
- The Completer infers the operand size from the Length field value and type code in the AtomicOp Request.
- The endian format used by AtomicOp Completers to read and write data at the target location is implementation specific, and permitted to be whatever the Completer determines to be appropriate for the target memory (e.g., little-endian, big-endian, etc.). See § [Section 2.2.2](#).
- If an AtomicOp Requester supports Address Translation Services (ATS), the Requester is permitted to use a Translated address in an AtomicOp Request only if the Translated address has appropriate access permissions. Specifically, the Read (R) and Write (W) fields must both be Set, and the Untranslated access only (U) field must be Clear. See § [Section 2.2.4.1](#).
- If a component supporting Access Control Services (ACS) supports AtomicOp routing or AtomicOp Requester capability, it handles AtomicOp Requests and Completions the same as with other Memory Requests and Completions with respect to ACS functionality.
- The No Snoop attribute is applicable and permitted to be Set with AtomicOp Requests, but atomicity must be guaranteed regardless of the No Snoop attribute value.
- The Relaxed Ordering attribute is applicable and permitted to be Set with AtomicOp Requests, where it affects the ordering of both the Requests and their associated Completions.
- Ordering requirements for AtomicOp Requests are similar to those for Non-Posted Write Requests. Thus, if a Requester wants to ensure that an AtomicOp Request is observed by the Completer before a subsequent Posted or Non-Posted Request, the Requester must wait for the AtomicOp Completion before issuing the subsequent Request.
- Ordering requirements for AtomicOp Completions are similar to those for Read Completions.
- Unless there’s a higher precedence error, an AtomicOp-aware Completer must handle a Poisoned AtomicOp Request as a Poisoned TLP Received error, and must also return a Completion with a Completion Status of Unsupported Request (UR). See § [Section 2.7.2.1](#). The value of the target location must remain unchanged.
- If the Completer of an AtomicOp Request encounters an uncorrectable error accessing the target location or carrying out the Atomic operation, the Completer must handle it as a Completer Abort (CA). The subsequent state of the target location is implementation specific.

- AtomicOp-aware Completers are required to handle any properly formed AtomicOp Requests with types or operand sizes they don't support as an Unsupported Request (UR). If the Length field in an AtomicOp Request contains an unarchitected value, the Request must be handled by an AtomicOp-aware Completer as a Malformed TLP. See § Section 2.2.7.
- If any Function in a Multi-Function Device supports AtomicOp Completer or AtomicOp routing capability, all Functions with Memory Space BARs in that device must decode properly formed AtomicOp Requests and handle any they don't support as an Unsupported Request (UR). Note that in such devices, Functions lacking AtomicOp Completer capability are forbidden to handle properly formed AtomicOp Requests as Malformed TLPS.
- If an RC has any Root Ports that support AtomicOp routing capability, all RCiEPs in the RC reachable by forwarded AtomicOp Requests must decode properly formed AtomicOp Requests and handle any they don't support as an Unsupported Request (UR).
- With an AtomicOp Request having a supported type and operand size, the AtomicOp-aware Completer is required either to carry out the Request or handle it as Completer Abort (CA) for any location in its target Memory Space. Completers are permitted to support AtomicOp Requests on a subset of their target Memory Space as needed by their programming model (see § Section 2.3.1). Memory Space structures defined or inherited by PCI Express (e.g., the MSI-X Table structure) are not required to be supported as AtomicOp targets unless explicitly stated in the description of the structure.
- For a Switch or an RC, when AtomicOp Egress Blocking is enabled in an Egress Port, and an AtomicOp Request targets going out that Egress Port, the Egress Port must handle the Request as an AtomicOp Egress Blocked error¹⁵³ (see § Figure 6-2) and must also return a Completion with a Completion Status of UR. If the severity of the AtomicOp Egress Blocked error is non-fatal, this case must be handled as an Advisory Non-Fatal Error as described in § Section 6.2.3.2.4.1.

6.15.3 Root Complex Support for AtomicOps §

RCs have unique requirements and considerations with respect to AtomicOp capabilities.

6.15.3.1 Root Ports with AtomicOp Completer Capabilities §

AtomicOp Completer capability for a Root Port indicates that the Root Port supports receiving at its Ingress Port AtomicOp Requests that target host memory or Memory Space allocated by a Root Port BAR. This is independent of any RCiEPs that have AtomicOp Completer capabilities.

If a Root Port implements any AtomicOp Completer capability for host memory access, it must implement all 32-bit and 64-bit AtomicOp Completer capabilities. Implementing 128-bit CAS Completer capability is optional.

If an RC has one or more Root Ports that implement AtomicOp Completer capability, the RC must ensure that host memory accesses to a target location on behalf of a given AtomicOp Request are performed atomically with respect to each host processor or device access to that target location range.

If a host processor supports atomic operations via its instruction set architecture, the RC must also ensure that host memory accesses on behalf of a given AtomicOp Request preserve the atomicity of any host processor atomic operations.

¹⁵³. Though an AtomicOp Egress Blocked error is handled by returning a Completion with UR Status, the error is not otherwise handled as an Unsupported Request. For example, it does not set the Unsupported Request Detected bit in the Device Status register.

6.15.3.2 Root Ports with AtomicOp Routing Capability §

As with other PCI Express Transactions, the support for peer-to-peer routing of AtomicOp Requests and Completions between Root Ports is optional and implementation dependent. If an RC supports AtomicOp routing capability between two or more Root Ports, it must indicate that capability in each associated Root Port via the AtomicOp Routing Supported bit in the Device Capabilities 2 register.

An RC is not required to support AtomicOp routing between all pairs of Root Ports that have the AtomicOp Routing Supported bit Set. An AtomicOp Request that would require routing between unsupported pairs of Root Ports must be handled as an Unsupported Request (UR), and reported by the “sending” Port.

The AtomicOp Routing Supported bit must be Set for any Root Port that supports forwarding of AtomicOp Requests initiated by host software or RCiEPs. The AtomicOp Routing Supported bit must be Set for any Root Ports that support forwarding of AtomicOp Requests received on their Ingress Port to RCiEPs.

6.15.3.3 RCs with AtomicOp Requester Capabilities §

An RC is permitted to implement the capability for either host software or RCiEPs to initiate AtomicOp Requests.

Software discovery of AtomicOp Requester capabilities is outside the scope of this specification.

If an RC supports software-initiated AtomicOp Requester capabilities, the specific mechanisms for how software running on a host processor causes the RC to generate AtomicOp Requests is outside the scope of this specification.

IMPLEMENTATION NOTE: GENERATING ATOMICOP REQUESTS VIA HOST PROCESSOR SOFTWARE §

If a host processor instruction set architecture (ISA) supports atomic operation instructions that directly correspond to one or more PCI Express AtomicOps, an RC might process the associated internal atomic transaction that targets PCI Express Memory Space much like it processes the internal read transaction resulting from a processor load instruction. However, instead of “exporting” the internal read transaction as a PCI Express Memory Read Request, the RC would export the internal atomic transaction as a PCI Express AtomicOp Request. Even if an RC uses the “export” approach for some AtomicOp types and operand sizes, it would not need to use this approach for all.

For AtomicOp types and operand sizes where the RC does not use the “export” approach, the RC might use an RC register-based mechanism similar to one where some PCI host bridges use CONFIG_ADDRESS and CONFIG_DATA registers to generate Configuration Requests. Refer to the [PCI] for details.

The “export” approach may permit a large number of concurrent AtomicOp Requests without becoming RC register limited. It may also be easier to support AtomicOp Request generation from user space software using this approach.

The RC register-based mechanism offers the advantage of working for all AtomicOp types and operand sizes even if the host processor ISA doesn’t support the corresponding atomic instructions. It might also support a polling mode for waiting on AtomicOp Completions as opposed to stalling the processor while waiting for a Completion.

6.15.4 Switch Support for AtomicOps §

If a Switch supports AtomicOp routing capability for any of its Ports, it must do so for all of them.

6.16 Dynamic Power Allocation (DPA) Capability §

A common approach to managing power consumption is through a negotiation between the device driver, operating system, and executing applications. Adding Dynamic Power Allocation for such devices is anticipated to be done as an extension of that negotiation, through software mechanisms that are outside of the scope of this specification. Some devices do not have a device specific driver to manage power efficiently. The DPA Capability provides a mechanism to allocate power dynamically for these types of devices. DPA is optional normative functionality applicable to Endpoint Functions that can benefit from the dynamic allocation of power and do not have an alternative mechanism. If supported, the Emergency Power Reduction State, over-rides the mechanisms listed here (see § Section 6.24).

The DPA Capability enables software to actively manage and optimize Function power usage when in the D0 state. DPA is not applicable to power states D1 - D3 therefore the DPA Capability is independently managed from the PCI-PM Capability.

DPA defines a set of power substates, each of which with an associated power allocation. Up to 32 substates [0..31] can be defined per Function. Substate 0, the default substate, indicates the maximum power the Function is ever capable of consuming.

Substates must be contiguously numbered from 0 to Substate_Max, as defined in § Section 7.9.12.2 . Each successive substate has a power allocation lower than or equal to that of the prior substate. For example, a Function with four substates could be defined as follows:

1. Substate 0 (the default) defines a power allocation of 25 Watts.
2. Substate 1 defines a power allocation of 20 Watts.
3. Substate 2 defines a power allocation of 20 Watts.
4. Substate 3 defines a power allocation of 10 Watts.

When the Function is initialized, it will operate within the power allocation associated with substate 0. Software is not required to progress through intermediate substates. Over time, software may dynamically configure the Function to operate at any of the substates in any sequence it chooses. Software is permitted to configure the Function to operate at any of the substates before the Function completes a previously initiated substate transition.

On the completion of the substate transition(s) the Function must compare its substate with the configured substate. If the Function substate does not match the configured substate, then the Function must begin transition to the configured substate. It is permitted for the Function to dynamically alter substate transitions on Configuration Requests instructing the Function to operate in a new substate.

In the prior example, software can configure the Function to transition to substate 4, followed by substate 1, followed by substate 3, and so forth. As a result, the Function must be able to transition between any substates when software configures the associated control field.

The Substate Control Enabled bit provides a mechanism that allows the DPA Capability to be used in conjunction with the software negotiation mechanism mentioned above. When Set, power allocation is controlled by the DPA Capability. When Clear, the DPA Capability is disabled, and the Function is not permitted to directly initiate substate transitions based on configuration of the Substate Control register field. At an appropriate point in time, software participating in the software negotiation mechanism mentioned above clears the bit, effectively taking over control of power allocation for the Function.

It is required that the Function respond to Configuration Space accesses while in any substate.

At any instant, the Function must never draw more power than it indicates through its Substate Status. When the Function is configured to transition from a higher power substate to a lower power substate, the Function's Substate Status must indicate the higher power substate during the transition, and must indicate the lower power substate after completing the transition. When the Function is configured to transition from a lower power substate to a higher power substate, the Function's Substate Status must indicate the higher power substate during the transition, as well as after completing the transition.

Due to the variety of applications and the wide range of maximum power required for a given Function, the transition time required between any substates is implementation specific. To enable software to construct power management policies (outside the scope of this specification), the Function defines two Transition Latency Values. Each of the Function substates associates its maximum Transition Latency with one of the Transition Latency Values, where the maximum Transition Latency is the time it takes for the Function to enter the configured substate from any other substate. A Function is permitted to complete the substate transition faster than the maximum Transition Latency for the substate.

6.16.1 DPA Capability with Multi-Function Devices §

Except as stated below, it is permitted for some or all Functions of a Multi-Function Device to implement a DPA Capability. The power allocation for the Multi-Function Device is the sum of power allocations set by the DPA Capability for each Function. It is permitted for the DPA Capability of a Function to include the power allocation for the Function itself as well as account for power allocation for other Functions that do not implement a DPA Capability. The association between multiple Functions for DPA is implementation specific and beyond the scope of this specification.

Power allocation for VFs is managed using their associated PF's DPA Capability, if implemented. VFs must not implement the Dynamic Power Allocation Capability.

6.17 TLP Processing Hints (TPH) §

TLP Processing Hints is an optional feature that provides hints in Request TLP headers to facilitate optimized processing of Requests that target Memory Space. These Processing Hints enable the system hardware (e.g., the Root Complex and/or Endpoints) to optimize platform resources such as system and memory interconnect on a per TLP basis. The TPH mechanism defines Processing Hints that provide information about the communication models between Endpoints and the Root Complex. Steering Tags are system-specific values used to identify a processing resource that a Requester explicitly targets. System software discovers and identifies TPH capabilities to determine the Steering Tag allocation for each Function that supports TPH.

6.17.1 Processing Hints §

The Requester provides hints to the Root Complex or other targets about the intended use of data and data structures by the host and/or device. The hints are provided by the Requester, which has knowledge of upcoming Request patterns, and which the Completer would not be able to deduce autonomously (with good accuracy). Cases of interest to distinguish with such hints include:

DWHR: Device writes then host reads soon

HWDR: Device reads data that the host is believed to have recently written

D*D*: Device writes/reads, then device reads/writes soon

Includes DWDW, DWDR, DRDW, DRDR

Bi-Directional: Data structure that is shared and has equal read/write access by host and device.

The usage models are mapped to the Processing Hint encodings as described in § Table 6-13 .

Table 6-13 Processing Hint Mapping §

| PH[1:0] (b) | Processing Hint | Usage Model |
|-------------|-------------------------------|---|
| 00 | Bi-directional data structure | Bi-Directional shared data structure |
| 01 | Requester | D*D* |
| 10 | Target | DWHR HWDR |
| 11 | Target with Priority | Same as target but with temporal ↑↑re-use↓↑↑reuse↑ priority |

6.17.2 Steering Tags §

Functions that intend to target a TLP towards a specific processing resource such as a host processor or system cache hierarchy require topological information of the target cache (e.g., which host cache). Steering Tags are system-specific values that provide information about the host or cache structure in the system cache hierarchy. These values are used to associate processing elements within the platform with the processing of Requests.

Software programmable Steering Tag values to be used are stored in an ST Table that is permitted to be located either in the TPH Requester Extended Capability structure (see § Section 7.9.13) or combined with the MSI-X Table (see § Section 7.7), but not in both locations for a given Function. When the ST Table is combined with the MSI-X Table, the 2 most significant bytes of the Vector Control register of each MSI-X Table entry are used to contain the Steering Tag value.

The choice of ST Table location is implementation specific and is discoverable by software. A Function that implements MSI-X is permitted to locate the ST Table in either location (see § Section 7.9.13.2). A Function that implements both MSI and MSI-X is permitted to combine the ST Table with the MSI-X Table and use it, even when MSI-X is disabled (i.e., when MSI is enabled). Each ST Table entry is 2 bytes. The size of the ST Table is indicated in the TPH Requester Extended Capability structure.

For some usage models the Steering Tags are not required or not provided, and in such cases a Function is permitted to use a value of all zeros in the ST field to indicate no ST preference. The association of each Request with an ST Table entry is device specific and outside the scope of this specification.

6.17.3 ST Modes of Operation §

The ST Table Location field in the TPH Requester Extended Capability structure indicates where (if at all) the ST Table is implemented by the Function. If an ST Table is implemented, software can program it with the system-specific Steering Tag values.

Table 6-14 ST Modes of Operation §

| ST Mode Select [2:0] (b) | ST Mode Name | Description |
|--------------------------|--------------|---|
| 000 | No ST Mode | The Function must use a value of all zeros for all Steering Tags. |

| ST Mode Select [2:0] (b) | ST Mode Name | Description |
|--------------------------|-----------------------|--|
| 001 | Interrupt Vector Mode | Each Steering Tag is selected by an MSI/MSI-X interrupt vector number. The Function is required to use the Steering Tag value from an ST Table entry that can be indexed by a valid MSI/MSI-X interrupt vector number. |
| 010 | Device Specific Mode | It is recommended for the Function to use a Steering Tag value from an ST Table entry, but it is not required. |
| All other encodings | Reserved | Reserved for future use. |

In the No ST Mode of operation, the Function must use a value of all zeros for each Steering Tag, enabling the use of Processing Hints without software-provided Steering Tags.

In the Interrupt Vector Mode of operation, Steering Tags are selected from the ST Table using MSI/MSI-X interrupt vector numbers. For Functions that have MSI enabled, the Function is required to select tags within the range specified by the Multiple Message Enable field in the MSI Capability structure. For Functions that have MSI-X enabled, the Function is required to select tags within the range of the MSI-X Table size. If the ST Table Size is smaller than the enabled range of interrupt vector numbers, the Function is permitted to either not use TPH for certain transactions, to use TPH with a Steering Tag of 0 or to use TPH with an implementation defined mechanism used to select a Steering Tag value from the ST Table. If the ST Table Size is larger than the enabled range of interrupt vector numbers, ST Table Entries corresponding to out of range interrupt vector numbers are ignored by the Function.

In the Device Specific Mode of operation, the assignment of the Steering Tags to Requests is device specific. The number of Steering Tags used by the Function is permitted to be different than the number of interrupt vectors allocated for the Function, irrespective of the ST Table location, and Steering Tag values used in Requests are not required to come from the architected ST Table.

A Function that is capable of generating TPH Requests is required to support the No ST Mode of operation. Support for other ST Modes of operation is optional. Only one ST Mode of operation can be selected at a time by programming ST Mode Select.

IMPLEMENTATION NOTE: ST TABLE PROGRAMMING §

To ensure that deterministic Steering Tag values are used in Requests, it is recommended that software either quiesce the Function or disable the TPH Requester capability during the process of performing ST Table updates. Failure to do so may result in non-deterministic values of ST values being used during ST Table updates.

6.17.4 TPH Capability §

TPH capabilities are optional normative. Each Function capable of generating Request TLPs with TPH is required to implement a TPH Requester Extended Capability structure. Functions that support processing of TLPs with TPH as Completers are required to indicate TPH Completer capability via the Device Capabilities 2 register. TPH is architected to be applied for transactions that target Memory Space, and is applicable for transaction flows between device-to-host, device-to-device and host-to-device. In each case for TPH to be supported, the Requester, Completer, and all intermediate routing elements must support the associated TPH capabilities.

Software discovers the Requester capabilities via the TPH Requester Extended Capability structure and Completer capabilities via the Device Capabilities 2 Register (see § Section 7.5.3.15). Software must program the TPH Requester Enable field in the TPH Requester Extended Capability structure to enable the Function to initiate Requests with TPH.

TPH only provides additional information to enable optimized processing of Requests that target Memory Space, so existing mechanisms and rules for managing Memory Space access (e.g., Bus Master Enable, Memory Space Enable, and Base Address registers) are not altered.

6.18 Latency Tolerance Reporting (LTR) Mechanism §

The Latency Tolerance Reporting (LTR) mechanism enables Endpoints to report their service latency requirements for Memory Reads and Writes to the Root Complex, so that power management policies for central platform resources (such as main memory, RC internal interconnects, and snoop resources) can be implemented to consider Endpoint service requirements. The LTR Mechanism does not directly affect Link power management or Switch internal power management, although it is possible that indirect effects will occur.

The implications of “latency tolerance” will vary significantly between different device types and implementations. When implementing this mechanism, it will generally be desirable to consider if service latencies impact functionality or only performance, if performance impacts are linear, and how much it is possible for the device to use buffering and/or other techniques to compensate for latency sensitivities.

The Root Complex is not required to honor the requested service latencies, but is strongly encouraged to provide a worst case service latency that does not exceed the latencies indicated by the LTR mechanism.

LTR support is discovered and enabled through reporting and control registers described in § Chapter 7. Software must not enable LTR in an Endpoint unless the Root Complex and all intermediate Switches indicate support for LTR. Note that it is not required that all Endpoints support LTR to permit enabling LTR in those Endpoints that do support it. When enabling the LTR mechanism in a hierarchy, devices closest to the Root Port must be enabled first.

If an LTR Message is received at a Downstream Port that does not support LTR or if LTR is not enabled, the Message must be treated as an Unsupported Request.

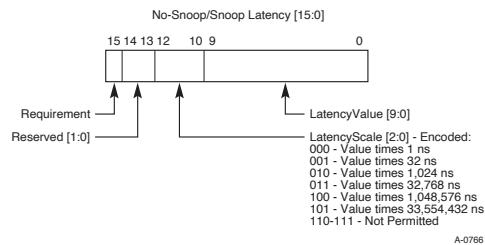


Figure 6-15 Latency Fields Format for LTR Messages §

No-Snoop Latency and Snoop Latency: As shown in § Figure 6-15, these fields include a Requirement bit that indicates if the device has a latency requirement for the given type of Request. If the Requirement bit is Set, the LatencyValue and LatencyScale fields describe the latency requirement. If the Requirement bit is Clear, there is no latency requirement and the LatencyValue and LatencyScale fields are ignored. With any LTR Message transmission, it is permitted for a device to indicate that a requirement is being reported for only no-snoop Requests, for only snoop Requests, or for both types of Requests. It is also permitted for a device to indicate that it has no requirement for either type of traffic, which it does by clearing the Requirement bit in both fields.

Each field also includes value and scale fields that encode the reported latency. Values are multiplied by the indicated scale to yield an absolute time value, expressible in a range from 1 ns to $2^{25} * (2^{10} - 1) = 34,326,183,936$ ns.

Setting the value and scale fields to all 0's indicates that the device will be impacted by any delay and that the best possible service is requested.

If a device doesn't implement or has no service requirements for a particular type of traffic, then it must have the Requirement bit clear for the associated latency field.

When directed to a non-D0 state by a Write to the PMCSR register, if a device's most recently transmitted LTR message (since the last DL_Down to DL_Up transition) reported one or both latency fields with any Requirement bit set, then it must send a new LTR Message with both of the Requirement bits clear prior to transitioning to the non-D0 state. If subsequently directed back to D0 by a Write to the PMCSR register, it is recommended that the device send a new LTR Message to re-establish its latency requirements after the transition back to D0 .

When the LTR Mechanism Enable bit is cleared, if a device's most recently sent LTR Message (since the last DL_Down to DL_Up transition) reported latency tolerance values with any Requirement bit set, then one of the following applies:

- If the bit was cleared due to a Configuration Write to the Device Control 2 Register , the device must send a new LTR Message with all the Requirement bits clear.
- If the bit was cleared due to an FLR , the device *MUST@FLIT* send a new LTR Message with all the Requirement bits clear.

When a Downstream Port goes to DL_Down status, any previous latencies recorded for that Port must be treated as invalid.

An LTR Message from a device reflects the tolerable latency from the perspective of the device, for which the platform must consider the service latency itself, plus the delay added by the use of Clock Power Management (CLKREQ#), if applicable. The service latency itself is defined as follows:

- When a device issues a Non-Posted Request, service latency of that Request is the delay from transmission of the last symbol of the Request TLP to the receipt of the first symbol of the first Completion TLP for that Request¹⁵⁴ .
- When a device issues one or more Posted Requests such that it cannot issue another Posted Request due to Flow Control backpressure, service latency of the blocked Request is the delay from the transmission of the last symbol of the previous Posted Request to the receipt of the first symbol of the DLLP returning the credit(s) that allows transmission of the blocked Request¹¹⁸ .

If Clock Power Management is used, then the platform implementation-dependent period between when a device asserts CLKREQ# and the device receives a valid clock signal constitutes an additional component of the platform service latency that must be comprehended by the platform when setting platform power management policy.

It is recommended that Endpoints transmit an LTR Message Upstream shortly after LTR is enabled, and subsequently when the Endpoint's service requirements change.

It is strongly recommended that Endpoints send no more than two LTR Messages within any 500 µs time period, except where required to by the specification. Downstream Ports must not generate an error if more than two LTR Messages are received within a 500 µs time period, and must properly handle all LTR messages regardless of the time interval between them.

Multi-Function Devices (MFDs) associated with an Upstream Port must transmit a “conglomerated” LTR Message Upstream according to the following rules:

- The acceptable latency values for the Message sent Upstream by the MFD must reflect the lowest values associated with any Function.

¹⁵⁴. For this definition, all of the symbols of a DLLP or TLP are included. For 8b/10b, the first and last symbols are framing symbols (SDP, STP or END, see § Section 4.2.1). For 128b/130b, the first symbol of a packet is the first symbol of a framing token (SDP or STP) and the last symbol of a packet is the last symbol of a CRC or LCRC (see § Section 4.2.2).

- It is permitted that the snoop and no-snoop latencies reported in the conglomerated Message are associated with different Functions.
- If none of the Functions report a requirement for a certain type of traffic (snoop/no-snoop), the Message sent by the MFD must not set the Requirement bit corresponding to that type of traffic.
- The MFD must transmit a new LTR Message Upstream when any Function of the MFD changes the values it has reported internally in such a way as to change the conglomerated value earlier reported by the MFD.

Switches must collect the Messages from Downstream Ports that have the LTR mechanism enabled and transmit a “conglomerated” Message Upstream according to the following rules:

- If a Switch supports the LTR feature, it must support the feature on its Upstream Port and all Downstream Ports.
- A Switch Upstream Port is permitted to transmit LTR Messages only when its LTR Mechanism Enable bit is Set or shortly after software clears its LTR Mechanism Enable bit as described earlier in this section.
- The acceptable latency values for the Message sent Upstream by the Switch must be calculated as follows:
 - If none of the Downstream Ports receive an LTR Message containing a requirement for a certain type of traffic (snoop/no-snoop), then any LTR Message sent by the Switch must not set the Requirement bit corresponding to that type of traffic.
 - Define LTRdnport[N] as the value reported in the LTR Message received at Downstream Port N, with these adjustments if applicable:
 - LTRdnport[N] is effectively infinite if the Requirement bit is clear or if a Not Permitted LatencyScale value is used
 - LTRdnport[N] must be 0 if the Requirement bit is 1 and the LatencyValue field is all 0's regardless of the LatencyScale value
 - Define LTRdnportMin as the minimum value of LTRdnport[N] across all Downstream Ports
 - Define Lswitch as all latency induced by the Switch
 - If Lswitch dynamically changes based on the Switch's operational mode, the Switch must not allow Lswitch to exceed 20% of LTRdnportMin, unless Lswitch for the Switch's lowest latency mode is greater, in which case the lowest latency state must be used
 - Calculate the value to transmit upstream, LTRconglomerated, as LTRdnportMin - Lswitch, unless this value is less than 0 in which case LTRconglomerated is 0
 - If LTRconglomerated is 0, both the LatencyValue and LatencyScale fields must be all 0's in the conglomerated LTR message
- A new LTR message must be transmitted Upstream if the conglomerated latencies are changed as a result of DL_Down invalidating the previous latencies recorded for that Port.
- If a Switch Downstream Port has the LTR Mechanism Enable bit cleared, the Latency Tolerance values recorded for that Port must be treated as invalid, and the latencies to be transmitted Upstream updated and a new conglomerated Message transmitted Upstream if the conglomerated latencies are changed as a result.
- A Switch must transmit an LTR Message Upstream when any Downstream ~~↓↑Port/~~
~~Function~~ ↑↓Port Function↑ changes the latencies it has reported in such a way as to change the conglomerated latency reported by the Switch.
- A Switch must not transmit LTR Messages Upstream unless triggered to do so by one of the events described above.

Errata: Base 6.3
B823△◀▶

The RC is permitted to delay processing of device Request TLPs provided it satisfies the device's service requirements.

When the latency requirement is updated during a series of Requests, it is required that the updated latency figure be comprehended by the RC prior to servicing a subsequent Request. In all cases the updated latency value must take effect

within a time period equal to or less than the previously reported latency requirement. It is permitted for the RC to comprehend the updated latency figure earlier than this limit.

IMPLEMENTATION NOTE: OPTIMAL USE OF LTR §

It is recommended that Endpoints transmit an updated LTR Message each time the Endpoint's service requirements change. If the latency tolerance is being reduced, it is recommended to transmit the updated LTR Message ahead of the first anticipated Request with the new requirement, allowing the amount of time indicated in the previously issued LTR Message. If the tolerance is being increased, then the update should immediately follow the final Request with the preceding latency tolerance value.

Typically, the Link will be in ASPM L1, and, if Clock Power Management (Clock PM) is supported, CLKREQ# will be deasserted, at the time an Endpoint reaches an internal trigger that causes the Endpoint to initiate Requests to the RC. The following text shows an example of how LTR is applied in such a case. Key time points are illustrated in § Figure 6-16.

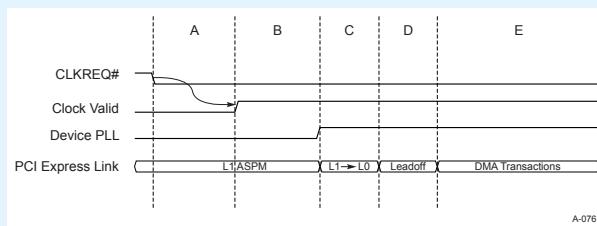


Figure 6-16 CLKREQ# and Clock Power Management §

Time A is a platform implementation-dependent period between when a device asserts CLKREQ# and the device receives a valid clock signal. This value will not exceed the latency in effect.

Time B is the device implementation-dependent period between when a device has a valid clock and it can initiate the retraining sequence to transition from L1 ASPM to L0.

Time C is the period during which the transition from L1 ASPM to L0 takes place

Time D for a Read transaction is the time between the transmission of the END symbol in the Request TLP to the receipt of the STP symbol in the Completion TLP for that Request. Time D for a Write transaction is the time between the transmission of the END symbol of the TLP that exhausts the FC credits to the receipt of the SDP symbol in the DLLP returning more credits for that Request type. This value will not exceed the latency in effect.

Time E is the period where the data path from the Endpoint to system memory is open, and data transactions are not subject to the leadoff latency.

The LTR latency semantic reflects the tolerable latency seen by the device as measured by one or both of the following:

Case 1: the device may or may not support Clock PM, but has not deasserted its CLKREQ# signal - The latency observed by the device is represented in § Figure 6-16 as the sum of times C and D.

Case 2: the device supports Clock PM and has deasserted CLKREQ#- The latency observed by the device is represented as the sum of times A, C, and D.

To effectively use the LTR mechanism in conjunction with Clock PM, the device will know or be able to measure times B and C, so that it knows when to assert CLKREQ#. The actual values of Time A, Time C, and Time D may vary dynamically, and it is the responsibility of the platform to ensure the sum will not exceed the latency.

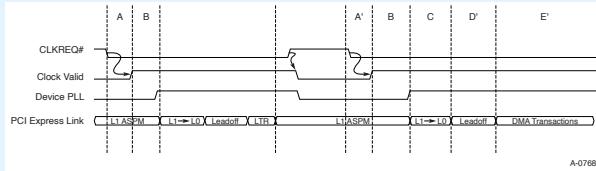


Figure 6-17 Use of LTR and Clock Power Management §

In a very simple model, an Endpoint may choose to implement LTR as shown in § Figure 6-17 . When an Endpoint determines that it is idle, it sends an LTR Message with the software configured maximum latency or the maximum latency the Endpoint can support.

When the Endpoint determines that it has a need to maintain sustained data transfers with the Root Complex, the Endpoint sends a new LTR Message with a shorter latency (at Time E). This LTR Message is sent prior to the next data flush by a time equal to the maximum latency sent before (the time between Time E and Time D'). In between Time E and Time A', the Endpoint can return to a low power state, while the platform transitions to a state where it can provide the shorter latency when the device next needs to transmit data.

Note that the RC may delay processing of device Request TLPs, provided it satisfies the device's service requirements. If, for example, an Endpoint connected to Root Port 1 reports a latency tolerance of 100 µs, and an Endpoint on Root Port 2 report a value of 30 µs, the RC might implement a policy of stalling an initial Request following an idle period from Root Port 1 for 70 µs before servicing the Request with a 30 µs latency, thus providing a perceived service latency to the first Endpoint of 100 µs. This RC behavior provides the RC the ability to batch together Requests for more efficient servicing.

It is possible that, after it is determined that the RC can service snoop and no-snoop Requests from all Endpoints within the maximum snoop and maximum no-snoop time intervals, this information may be communicated to Endpoints by updating the Max Snoop LatencyValue, Max Snoop LatencyScale and Max No-Snoop LatencyValue, Max No-Snoop LatencyScale fields. The intention of this communication would be to prevent Endpoints from sending needless LTR updates.

When an Endpoint's LTR value for snoop Requests changes to become larger (looser) than the value indicated in the Max Snoop LatencyValue/Scale fields, it is recommended that the Endpoint send an LTR message with the snoop LTR value indicated in the Max Snoop LatencyValue/Scale fields. Likewise, when an Endpoint's LTR value for no-snoop Requests changes to become larger (looser) than the value indicated in the Max No-Snoop LatencyValue/Scale fields, it is recommended that the Endpoint send an LTR message with the no-snoop LTR value indicated in the Max No-Snoop LatencyValue/Scale fields.

It is recommended that Endpoints buffer Requests as much as possible, and then use the full Link bandwidth in bursts that are as long as the Endpoint can practically support, as this will generally lead to the best overall platform power efficiency.

Note that LTR may be enabled in environments where not all Endpoints support LTR, and in such environments, Endpoints that do not support LTR may experience suboptimal service.

6.19 Optimized Buffer Flush/Fill (OBFF) Mechanism §

The Optimized Buffer Flush/Fill (OBFF) Mechanism enables a Root Complex to report to Endpoints (throughout a hierarchy) time windows when the incremental platform power cost for Endpoint bus mastering and/or interrupt activity is relatively low. Typically this will correspond to time that the host CPU(s), memory, and other central resources

associated with the Root Complex are active to service some other activity, for example the operating system timer tick. The nature and determination of such windows is platform/implementation specific.

An OBFF indication is a hint - Functions are still permitted to initiate bus mastering and/or interrupt traffic whenever enabled to do so, although this will not be optimal for platform power and should be avoided as much as possible.

OBFF is indicated using either the WAKE# signal or a message (see § Section 2.2.8.9). The message is to be used exclusively on interconnects where the WAKE# signal is not available. WAKE# signaling of OBFF or CPU Active must only be initiated by a Root Port when the system is in an operational state, which in an ACPI compliant system corresponds to the S0 state. Functions that are in a non-D0 state must not respond to OBFF or CPU Active signaling.

OBFF messages use Message Routing 100b, "Local - Terminate at Receiver" (see § Table 2-20), and are only permitted to be transmitted in the Downstream direction. There are multiple OBFF events distinguished. When using the OBFF Message, the OBFF Code field (see [§ Figure 2-65](#) and [§ Figure 2-66](#)) is used to distinguish between different OBFF cases:

1111b “CPU Active”

System fully active for all Device actions including bus mastering and interrupts

0001b “OBFF”

System memory path available for Device memory read/write bus master activities

0000b “Idle”

System in an idle, low power state

Others

All other codes are Reserved.

These codes correspond to various assertion patterns of WAKE# when using WAKE# signaling, as shown in [§ Figure 6-18](#). There is one negative-going transition when signaling OBFF and two negative going transitions each time CPU Active is signaled. The electrical parameters required when using WAKE# are defined in the WAKE# Signaling section of [\[CEM-2.0\]](#) (or later).

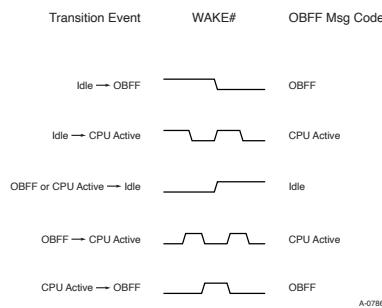


Figure 6-18 Codes and Equivalent WAKE# Patterns [§](#)

When an OBFF Message is received that indicates a Reserved code, the Receiver, if OBFF is enabled, must treat the indication as a “CPU Active” indication.

An OBFF Message received at a Port that does not implement OBFF or when OBFF is not enabled must be handled as an Unsupported Request (UR). This is a reported error associated with the receiving Port (see § Section 6.2). If a Port has OBFF enabled using WAKE# signaling, and that Port receives an OBFF Message, the behavior is undefined.

OBFF indications reflect central resource power management state transitions, and are signaled using WAKE# when this is supported by the platform topology, or using a Message when WAKE# is not available. OBFF support is discovered and enabled through reporting and control registers described in § Chapter 7.. Software must not enable OBFF in an Endpoint unless the platform supports delivering OBFF indications to that Endpoint.

When the platform indicates the start of a CPU Active or OBFF window, it is recommended that the platform not return to the Idle state in less than 10 µs. It is permitted to indicate a return to Idle in advance of actually entering platform idle, but it is strongly recommended that this only be done to prevent late Endpoint activity from causing an immediate exit from the idle state, and that the advance time be as short as possible.

It is recommended that Endpoints not assume CPU Active or OBFF windows will remain open for any particular length of time.

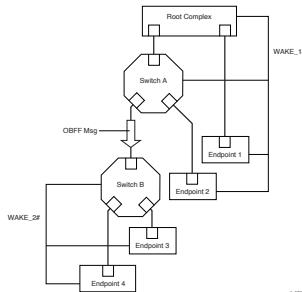


Figure 6-19 Example Platform Topology Showing a Link Where OBFF is Carried by Messages §

§ Figure 6-19 shows an example system where it is necessary for a Switch (A) to translate OBFF indications received using WAKE# into OBFF Messages, which in this case are received by another Switch (B) and translated back to using WAKE# signaling. A HwInit configuration mechanism (set by hardware or firmware) is used to identify cases such as shown in this example (where the link between Switch A and Switch B requires the use of OBFF Messages), and system firmware/software must configure OBFF accordingly.

When a Switch is configured to use OBFF Message signaling at its Upstream Port and WAKE# at one or more Downstream Ports, or vice-versa, when enabled for OBFF, the Switch is required to convert all OBFF indications received at the Upstream Port into the appropriate form at the Downstream Port(s).

When using WAKE#, the enable for any specific Root Port enables the global use of WAKE# unless there are multiple WAKE# signals, in which case only the associated WAKE# signals are affected. When using Message signaling for OBFF, the enable for a particular Root Port enables transmission of OBFF messages from that Root Port only. To ensure OBFF is fully enabled in a platform, all Root Ports indicating OBFF support must be enabled for OBFF. It is permitted for system firmware/software to selectively enable OBFF, but such enabling is beyond the scope of this specification.

To minimize power consumption, system firmware/software is strongly recommended to enable Message signaling of OBFF only when WAKE# signaling is not available for a given link.

OBFF signaling using WAKE# must only be reported as supported by all components connected to a Switch if it is a shared WAKE# signal. In these topologies it is permitted for software to enable OBFF for components connected to the Switch even if the Switch itself does not support OBFF.

It is permitted, although not encouraged, to indicate the same OBFF event more than once in succession.

When a Switch is propagating OBFF indications Downstream, it is strongly encouraged to propagate all OBFF indications. However, especially when using Messages, it may be necessary for the Switch to discard or collapse OBFF indications. It is permitted to discard and replace an earlier indication of a given type when an indication of the same or a different type is received.

Downstream Ports can be configured to transmit OBFF Messages in two ways, which are referred to as Variation A and Variation B. For Variation A, the Port must transmit the OBFF Message if the Link is in the L0 state, but discard the Message when the Link is in the Tx_L0s or L1 state. This variation is preferred when the Downstream Port leads to Devices that are expected to have communication requirements that are not time-critical, and where Devices are expected to signal a non-urgent need for attention by returning the Link state to L0. For Variation B, the Port must

transmit the OBFF Message if the Link is in the L0 state, or, if the Link is in the Tx_L0s or L1 state, it must direct the Link to the L0 state and then transmit the OBFF Message. This variation is preferred when the Downstream Port leads to devices that can benefit from timely notification of the platform state.

When initially configured for OBFF operation, the initial assumed indication must be the CPU Active state, regardless of the logical value of the WAKE# signal, until the first transition is observed.

When enabling Ports for OBFF, it is recommended that all Upstream Ports be enabled before Downstream Ports, and Root Ports must be enabled after all other Ports have been enabled. For hot pluggable Ports this sequence will not generally be possible, and it is permissible to enable OBFF using WAKE# to an unconnected hot pluggable Downstream Port. It is recommended that unconnected hot pluggable Downstream Ports not be enabled for OBFF message transmission.

IMPLEMENTATION NOTE: OBFF CONSIDERATIONS FOR ENDPOINTS §

It is possible that during normal circumstances, events could legally occur that could cause an Endpoint to misinterpret transitions from an Idle window to a CPU Active window or OBFF window. For example, a non-OBFF Endpoint could assert WAKE# as a wakeup mechanism, masking the system's transitions of the signal. This could cause the Endpoint to behave in a manner that would be less than optimal for power or performance reasons, but should not be unrecoverable for the Endpoint or the host system.

In order to allow an Endpoint to maintain the most accurate possible view of the host state, it is recommended that the Endpoint place its internal state tracking logic in the CPU Active state when it receives a request that it determines to be host-initiated, and at any point where the Endpoint has a pending interrupt serviced by host software.

6.20 PASID §

In Non-Flit Mode, the PASID TLP Prefix is an End-End TLP Prefix as defined in § Section 2.2.1 . Layout of the PASID TLP Prefix is shown in § Figure 6-20 and § Table 6-15 .

In Flit Mode, the PASID, when present, is included in OHC-A1 or OHC-A4 .

When a PASID is present, the PASID value, in conjunction with the Requester ID, identifies the Process Address Space ID associated with the Request. Each Function has a distinct set of PASID values. PASID values used by one Function are unrelated to PASID values used by any other Function.

PASID is permitted only for specific types of TLPs:

- Requests that include an Address with Address Type (AT) of Untranslated or Translated (see § Section 2.2.4.1).
- Address Translation Requests (i.e., MRd with AT =01b), ATS Invalidate Request Messages, Page Request Messages, Address routed messages in Flit Mode, and PRG Response Messages (see § Section 10.1.3).

PASID is not permitted on any other type of TLP.

6.20.1 Managing PASID Usage §

Usage of PASID is permitted only when specifically enabled.

For Endpoint Functions (including Root Complex Integrated Devices), the following rules apply:

- A Function is not permitted to send and receive TLPs with a PASID unless PASID Enable is Set (see § Section 7.8.9.3):
- A Function is not permitted to generate Requests using Translated Addresses and a PASID unless both PASID Enable and Translated Requests with PASID Enable are Set.
- A Function must have a mechanism for associating use of a PASID with a particular Function context. This mechanism is outside the scope of this specification.
- A Function must have a mechanism to request that it gracefully stop using a specific PASID. This mechanism is device specific but must satisfy the following rules:
 - A Function may support a limited number of simultaneous PASID stop requests. Software should defer issuing new stop requests until older stop requests have completed.
 - A stop request in one Function must not affect operation of any other Function.
 - A stop request must not affect operation of any other PASID within the Function.
 - A stop request must not affect operation of transactions that are not associated with a PASID.
 - When the stop request mechanism indicates completion, the Function has:
 - Stopped queuing new Requests for this PASID.
 - Completed all Non-Posted Requests **↑↑and UIO Requests↑** associated with this PASID.
 - Flushed to the host all Posted Requests addressing host memory in all TCs that were used by the PASID. The mechanism used for this is device specific (for example: a non-relaxed Posted Write to host memory or a processor read of the Function can flush TC0; a zero length read to host memory can flush non-zero TCs).
 - Optionally flushed all Peer-to-Peer Posted Requests to their destination(s). The mechanism used for this is device specific.
 - Complied with additional rules described in Address Translation Services (§ Chapter 10.) if Address Translations or Page Requests were issued on the behalf of this PASID.

Errata: Base 6.3
B834△◀▷

For Root Complexes, the following rules apply:

- A Root Complex must have an implementation specific mechanism for indicating support for PASID.
- A Root Complex that supports PASID must have an implementation specific mechanism for enabling them. By default usage of PASID is disabled.
- A Root Complex that supports PASID may optionally have an implementation specific mechanism for enabling them at a finer granularity than the entire Root Complex (e.g., distinct enables for a specific Root Port, Requester ID, Bus Number, Requester ID, or Requester ID/PASID combination).

6.20.2 PASID Information Layout §

In Flit Mode, the PASID information is contained in OHC-A1 or OHC-A4. In Non-Flit Mode, the PASID information is contained in a PASID TLP Prefix.

6.20.2.1 PASID TLP Prefix - Non-Flit Mode §

A TLP may contain at most one PASID TLP Prefix.

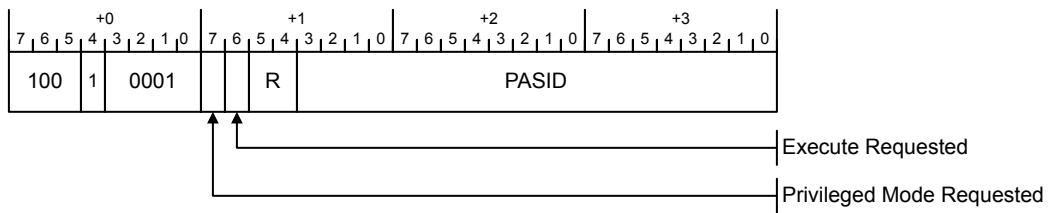


Figure 6-20 PASID TLP Prefix §

Table 6-15 PASID TLP Prefix §

| Bits | Description |
|-------------------------------|---|
| Byte 0 bits 7:5 | 100b - indicating TLP Prefix |
| Byte 0 bit 4 | 1b - indicating End-End TLP Prefix |
| Byte 0 bits 3:0 | 0001b - indicating PASID TLP Prefix |
| Byte 1 bit 7 | Privileged Mode Requested - If Set indicates a Privileged Mode entity in the Endpoint is issuing the Request, If Clear, indicates a Non-Privileged Mode entity in the Endpoint is issuing the Request. Usage of this bit is specified in § Section 6.20.2.4 . |
| Byte 1 bit 6 | Execute Requested - If Set indicates the Endpoint is requesting Execute Permission. If Clear, indicates the Endpoint is not requesting Execute Permission. Usage of this bit is specified in § Section 6.20.2.3 . |
| Byte 1 bit 3: Byte 3 bit 0 | Process Address Space ID (PASID) - This field contains the PASID value associated with the TLP. Usage of this field is defined in § Section 6.20.2.2 . |

6.20.2.2 PASID field (Flit Mode and Non-Flit Mode) §

The PASID field identifies the user process associated with a Request.

The PASID field is 20 bits wide. Endpoints and Root Complexes need not support the entire range of the field. For Endpoints, the Max PASID Width field indicates the supported range of PASID values (§ Section 7.8.9.2). For Root Complexes, an implementation specific mechanism is used to provide this information.

Endpoints are not permitted to send TLPs with a PASID unless the PASID Enable bit (§ Section 7.8.9.3) is Set. Endpoints that support PASID must signal Unsupported Request (UR) when they receive a TLP with a PASID and the PASID Enable bit is Clear.

Root Complexes may optionally support TLPs with a PASID. The mechanism used to detect whether a Root Complex supports PASID is implementation specific.

For Endpoints, the following rules apply:

- The Endpoint is not permitted to send TLPs with a PASID value greater than or equal to $2^{\text{Max PASID Width}}$.
- The Endpoint is optionally permitted to signal an error when it receives a Request with a PASID value greater than or equal to $2^{\text{Max PASID Width}}$. This is an Unsupported Request error associated with the Receiving Port (see § [Section 6.2](#)).

For Root Complexes, the following rules apply:

- A Root Complex is not permitted to send a TLP with a PASID value greater than it supports.
- A Root Complex is optionally permitted to signal an error when it receives a Request with a PASID value greater than it supports. This is an Unsupported Request error associated with the Receiving Port (see § [Section 6.2](#)).

For Completers, the following rules apply:

- For Untranslated Memory Requests, the PASID value and the Untranslated Address are both used in determining the Translated Address used in satisfying the Request.

For address translation related TLPs, usage of this field is defined in [Address Translation Services](#) (§ [Chapter 10](#)).

IMPLEMENTATION NOTE: PASID WIDTH HOMOGENEITY §

The PASID value is unique per Function and thus the original intent was that the width of the PASID value supported by that Function could be based on the needs of that Function. However, current system software typically does not follow that model and instead uses the same PASID value in all Functions that access a specific address space. To enable this, system software will typically ensure a common system PASID width for Root Complex and persistent translation agents. Such system software will typically disable ATS on any hot plugged Endpoint Functions or translation agents reporting PASID width support which is less than that of the common system PASID width.

The Root Complex, Endpoints, and translation agents, are often implemented independently of system software, therefore it is strongly recommended that hardware implement the maximum width of 20 bits to ensure interoperability with system software.

Endpoints may, in an implementation specific way, be able to map the 20 bit system PASID to an internal representation carrying a smaller width. If this is done, it is critical that the Endpoint do so without impacting system software, which has no mechanism to differentiate such implementation from those that implement the full 20 bit width natively.

6.20.2.3 Execute Requested §

If the Execute Requested bit is Set, the Endpoint is requesting permission for the Endpoint to Execute instructions in the memory range associated with this request. The meaning of Execute permission is outside the scope of this specification.

Endpoints are not permitted to send TLPs with the Execute Requested bit Set unless the Execute Permission Supported bit (§ [Section 7.8.9.2](#)) and the Execute Permission Enable bit (§ [Section 7.8.9.3](#)) are both Set.

For Root Complexes, the following rules apply:

- Support for Execute Requested by the Root Complex is optional. The mechanism used to determine whether a Root Complex supports Execute Requested is implementation specific.
 - It is strongly recommended that Root Complexes not support Execute Requested.
- A Root Complex that supports the Execute Requested bit must have an implementation specific mechanism to enable it to use the bit. This mechanism must default to disabled.
- A Root Complex that supports the Execute Requested bit is permitted to have an implementation specific mechanism to enable use of the bit at a finer granularity (e.g., for a specific Root Port, for a specific Bus Number, for a specific Requester ID, or for a specific Requester ID/PASID combination), and its default value is implementation specific.

For Completers, the following rules apply:

- Completers have a concept of an effective value of the bit. For a given Request, if the Execute Requested bit is supported and its usage is enabled for the Request, the effective value of the bit is the value in the Request; otherwise the effective value of the bit is 0b.
- For Untranslated Memory Read Requests, Completers use the effective value of the bit as part of the protection check. If this protection check fails, Completers treat the Request as if the memory was not mapped.
- For Memory Requests, other than an Untranslated Memory Read Request, the bit is Reserved.
For address translation related TLPs, usage of this bit is defined in Address Translation Services (§ Chapter 10.).

6.20.2.4 Privileged Mode Requested §

If Privileged Mode Requested is Set, the Endpoint is issuing a Request that targets memory associated with Privileged Mode. If Privileged Mode Requested is Clear, the Endpoint is issuing a Request that targets memory associated with Non-Privileged Mode.

The meaning of Privileged Mode and Non-Privileged Mode and what it means for an Endpoint to be operating in Privileged or Non-Privileged Mode depends on the protection model of the system and is outside the scope of this specification.

Endpoints are not permitted to send a TLP with the Privileged Mode Requested bit Set unless both the Privileged Mode Supported bit (§ Section 7.8.9.2) and the Privileged Mode Enable bit (§ Section 7.8.9.3) are Set.

For Root Complexes, the following rules apply:

- Support for the Privileged Mode Requested bit by the Root Complex is optional. The mechanism used to determine whether a Root Complex supports the Privileged Mode Requested bit is implementation specific.
- A Root Complex that supports the Privileged Mode Requested bit should have an implementation specific mechanism to enable it to use the bit.
- A Root Complex that supports the Privileged Mode Requested bit may have an implementation specific mechanism to enable use of the bit at a finer granularity (e.g., for a specific Root Port, for a specific Bus Number, for a specific Requester ID, or for a specific Requester ID/PASID combination).

For Completers, the following rules apply:

- Completers have the concept of an effective value of the bit. For a given Request, if the Privileged Mode Requested bit is supported and its usage is enabled for the Request, the effective value of the bit is the value in the Request; otherwise the effective value of the bit is the 0b.
- For Untranslated Memory Requests, Completers use the effective value of the bit as part of its protection check. If this protection check fails, Completers treat the Request as if the memory was not mapped.

- For address translation related TLPs, usage of this bit is defined in [Address Translation Services \(§ Chapter 10.\)](#).

6.21 Precision Time Measurement (PTM) Mechanism §

6.21.1 Introduction §

Precision Time Measurement (PTM) enables precise coordination of events across multiple components with independent local time clocks. Ordinarily, such precise coordination would be difficult given that individual time clocks have differing notions of the value and rate of change of time. To work around this limitation, PTM enables components to calculate the relationship between their local times and a shared PTM Master Time: an independent time domain associated with a PTM Root.

Enhanced Precision Time Measurement (ePTM) places additional requirements on PTM Devices. Support for ePTM is indicated by the ePTM Capable bit.

PTM defines the following:

- PTM Requester - A Function capable of using PTM as a consumer associated with an Endpoint or an Upstream Port.
- PTM Responder - A Function capable of using PTM to supply PTM Master Time associated with a Port or an RCRB.
- Time Source - A local clock associated with a PTM Responder.
- PTM Root - The source of PTM Master Time for a PTM Hierarchy. A PTM Root must also be a Time Source and is typically also a PTM Responder.

Each PTM Root supplies a single PTM Master Time to all of the PTM Hierarchy: a set of PTM Requesters associated with a single PTM Root.

§ Figure 6-21 illustrates some example system topologies using PTM. These are only illustrative examples, and are not intended to imply any limits or requirements.

Base 6.4 vs Base 6.3

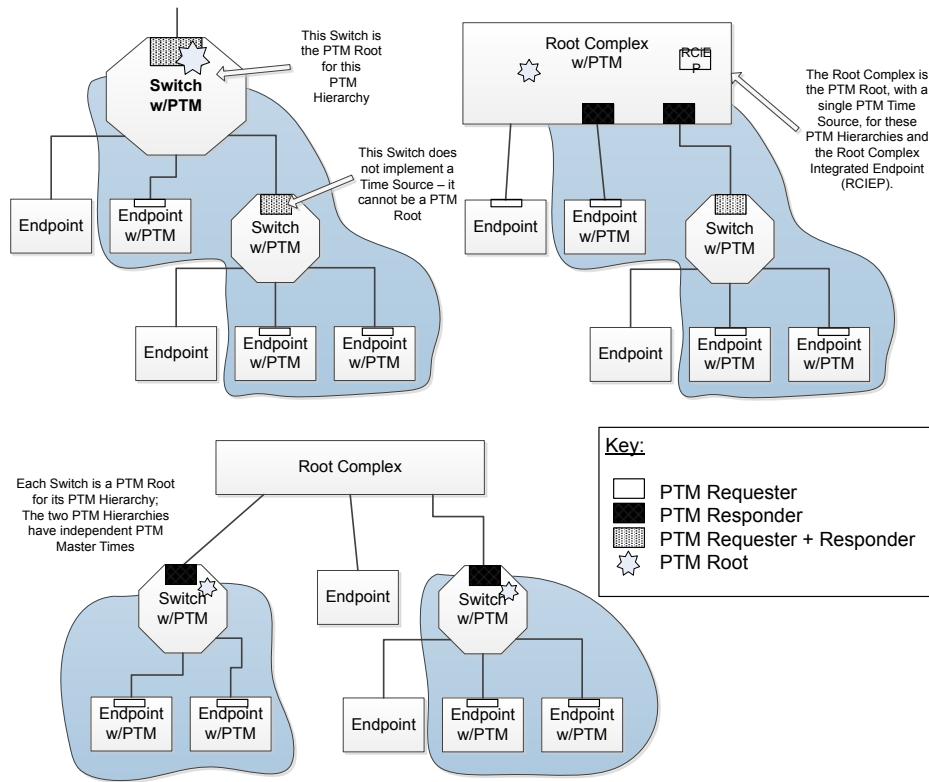


Figure 6-21 Example System Topologies using PTM §

IMPLEMENTATION NOTE: PTM AND RETIMERS §

PCIe Retimers can impact PTM accuracy by introducing asymmetric link delays. Retimers designed to maintain symmetric link delays will enable the best PTM accuracy. The larger and more variable the asymmetry, the greater the impact to PTM. Consult the manufacturer's documentation to determine the suitability of a Retimer implementation for use with PTM.

6.21.2 PTM Link Protocol §

When using PTM between two components on a Link, the Upstream Port, which acts on behalf of the PTM Requester, sends PTM Requests to the Downstream Port on the same Link, which acts on behalf of the PTM Responder.

Base 6.4 vs Base 6.3

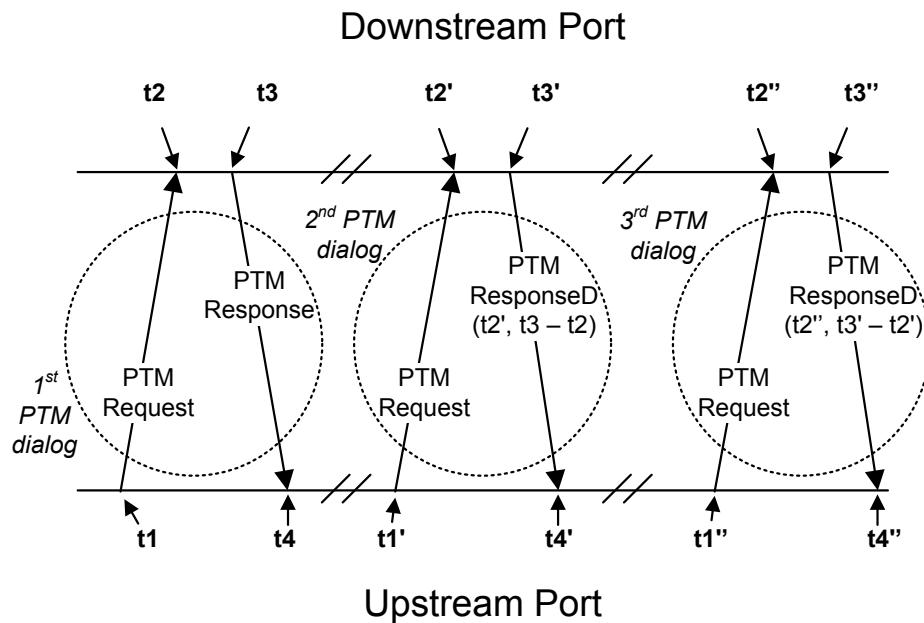


Figure 6-22 Precision Time Measurement Link Protocol §

§ Figure 6-22 illustrates the PTM link protocol. The points t_1 , t_2 , t_3 , and t_4 in the above diagram represent timestamps captured locally by each Port as they transmit and receive PTM Messages. The component associated with each Port stores these timestamps from the 1st PTM dialog in internal registers for use in the 2nd PTM dialog, and so on for subsequent PTM dialogs.

The Upstream Port, on behalf of the PTM Requester, initiates the PTM dialog by transmitting a PTM Request message.

The Downstream Port, on behalf of the PTM Responder, has knowledge of or access (directly or indirectly) to the PTM Master Time.

During each dialog, the Downstream Port populates the PTM ResponseD message based on timestamps stored during previous PTM dialogs, as defined in § Section 6.21.3.2.

Once each component has historical timestamps from the preceding dialog, the component associated with the Upstream Port can combine its timestamps with those passed in the PTM ResponseD message to calculate the PTM Master Time using the following formula:

$$\text{PTM Master Time at } t_1 = t_2 - \frac{\left((t_4 - t_1) - (t_3 - t_2) \right)}{2}$$

Equation 6-2 PTM Master Time §

The values t_1 , t_2 , t_3 , t_4 , and t_2' indicate the timestamps captured during the PTM dialog as illustrated in § Figure 6-22 .

PTM capable components would typically record the results of these timestamp calculations, and may make them available to software via implementation specific means. Herein, this document refers to this resultant timing information as the component's "PTM context".

For a Switch implementing PTM, the time synchronization mechanism(s) within the Switch itself are implementation specific.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: PTM THEORY AND OPERATION §

The timestamps captured during the PTM dialogs enable the calculation of the timing relationship between the PTM Requester and PTM Responder. The value $(t_3 - t_2)$ measures the time consumed by the PTM Responder for a given PTM dialog. The time $(t_4 - t_1)$ is the time from request to response. Therefore $((t_4 - t_1) - (t_3 - t_2))$ effectively gives the round trip message transit time between the two components, and that quantity divided by 2 approximates the Link delay - the time difference between t_1 and t_2 . It is assumed that the Link transit times from PTM Requester to PTM Responder and back again are symmetric, which is typically a good assumption (see also the Implementation Note on PTM Timestamp Capture Mechanisms).

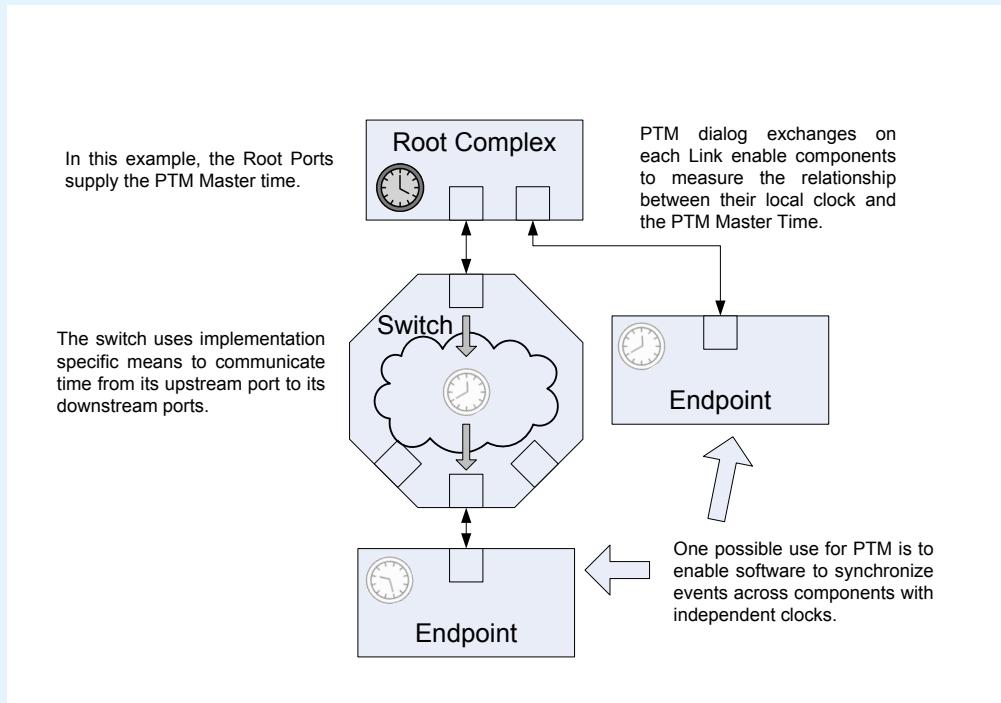


Figure 6-23 Precision Time Measurement Example §

§ Figure 6-23 illustrates a simple device hierarchy employing PTM. Each Upstream Port initiates PTM dialogs to establish the relationship between its local time and the PTM Master Time provided by the Root Port.

In this example, the Switch initiates PTM dialogs on its Upstream Port to obtain the PTM Master Time for use in fulfilling PTM Request Message received at its Downstream Ports. This Switch employs implementation specific means to communicate the PTM Master Time from its Upstream Port to its Downstream Ports.

PTM capable components can make their PTM context available for inspection by software, enabling software to translate timing information between local times and PTM Master Time. In turn, this capability enables software to coordinate events across multiple components with very fine precision.

Similarly, it is strongly recommended that platforms implementing PTM also make the PTM Master Time available to software.

6.21.3 Configuration and Operational Requirements §

Software must not have the PTM Enable bit Set in the PTM Control register on a Function associated with an Upstream Port unless the associated Downstream Port on the Link already has the PTM Enable bit Set in its associated PTM Control register.

PTM support by a Function is indicated by the presence of a PTM Extended Capability structure. It is not required that all Endpoints in a hierarchy support PTM, and it is not required for software to enable PTM in all Endpoints that do support it.

If a PTM Message is received by a Port that does not support PTM, or by a Downstream Port when the PTM Enable bit is clear, the Message must be treated as an Unsupported Request. This is a reported error associated with the Receiving Port (see § Section 6.2). A properly formed PTM Response received by an Upstream Port that supports PTM, but for which the PTM Enable bit is clear, must be silently discarded.

As observed through PTM, the PTM Master Time must satisfy the following behavioral requirements:

- Time values must be monotonic, and strictly increasing.
- The perceived granularity must be no greater than the value reported in the Local Clock Granularity field of the PTM Capability Register.
- The perceived time must start no later than when the PTM Root processes its first PTM Request Message.

Referring to § Figure 6-22, the following rules define timestamp capture:

- A PTM Requester must update its stored t1 timestamp when transmitting a PTM Request Message, even if that transmission is a replay.
- A PTM Responder must update its stored t2 timestamp when receiving a PTM Request Message, even if received TLP is a duplicate.
- A PTM Responder must update its stored t3 timestamp when transmitting a PTM Response or ResponseD Message, even if that transmission is a replay.
- A PTM Requester must update its stored t4 timestamp when receiving a PTM Response Message, even if received TLP is a duplicate.

In NFM, Timestamps must be based on the STP Symbol or Token that frames the TLP, as if observing the first bit of that Symbol or Token at the Port's pins.

In FM, a single Flit is permitted to include zero or one PTM Messages, and Timestamps must be based on the Flit_Marker (see § Section 4.2.3.4.2) that has the PTM Message contained in this Flit bit Set, as if observing the Flit_Marker Indicator bit at the Port's pins. Typically this will require an implementation specific adjustment to compensate for the inability to directly measure the time at the actual pins, as the time will commonly be measured at some internal point in the Rx or Tx path. The accuracy and consistency of this measurement are not bounded by this specification, but it is strongly recommended that the highest practical level of accuracy and consistency be achieved.

As illustrated in § Figure 2-68, the bytes within the Propagation Delay[31:0] field are such that:

- Data Byte 0 contains Propagation Delay [31:24] (most significant byte)
- Data Byte 1 contains Propagation Delay [23:16]
- Data Byte 2 contains Propagation Delay [15:8]
- Data Byte 3 contains Propagation Delay [7:0] (least significant byte)

All implementations compliant to this document are required to follow the above interpretation (Propagation Delay interpretation A). Due to ambiguity in earlier versions of this document, some implementations made this interpretation (Propagation Delay interpretation B):

- Data Byte 0 contains Propagation Delay [7:0] (least significant byte)
- Data Byte 1 contains Propagation Delay [15:8]
- Data Byte 2 contains Propagation Delay [23:16]
- Data Byte 3 contains Propagation Delay [31:24] (most significant byte)

To allow implementations using interpretation A to interoperate with implementations using interpretation B the PTM Propagation Delay Adaptation Capability can be used. For an Upstream Port, this capability applies to the interpretation of received PTM ResponseD Messages , for a Downstream Port, this capability applies to the interpretation of transmitted PTM ResponseD Messages . Support for the PTM Propagation Delay Adaptation Capability is indicated by Setting the PTM Propagation Delay Adaptation Capable bit in the PTM Capability Register . When supported, the Port must use interpretation A, unless the PTM Propagation Delay Adaptation Interpretation B bit in the Link Control Register is Set, in which case the Port changes to interpretation B. For a Switch, if the PTM Propagation Delay Adaptation Capable bit is Set, then all Ports of the Switch must support the PTM Propagation Delay Adaptation Capability , and each Switch Port must be controlled independently by the value of the PTM Propagation Delay Adaptation Interpretation B bit in the Link Control Register for the Port.

It is strongly recommended that software not enable PTM on a link until it has first assured, either by means of the PTM Propagation Delay Adaptation Capability or in an implementation specific manner, that both Ports on the Link are able to compatibly interpret the Propagation Delay value.

6.21.3.1 PTM Requester Role §

- Support for the PTM Requester role is indicated by setting the PTM Requester Capable bit in the PTM Capability Register .
- PTM Requesters are permitted to request PTM Master Time only when PTM is enabled. The mechanism for directing a PTM Requester to issue such a request is implementation specific.
 - Upstream Ports obtain PTM Master Time via PTM dialogs as described in § Section 2.2.8.10 .
 - The mechanism by which RCiEPs request PTM Master Time is implementation specific.
- Once having issued a PTM Request Message, the Upstream Port must not issue another PTM Request Message prior to the receipt of a PTM Response Message, PTM ResponseD Message, Reset, or the passage of 100 µs without a corresponding PTM Message from the Downstream Port.
- Upon receiving a PTM Response, the Upstream Port must wait at least 1 µs before issuing another PTM Request Message.
- For Multi-Function Devices (MFDs) containing multiple PTM Requesters, the Upstream Port associated with that MFD must issue a single PTM dialog during each PTM context refresh. PTM Requesters within the MFD maintain their individual PTM contexts using this one, Device-wide PTM dialog. The mechanism for refreshing multiple PTM contexts from one PTM dialog is implementation specific.
- An Upstream Port *MUST@FLIT* invalidate its internal PTM context when any of the following occur. If ePTM is supported, then an Upstream Port must invalidate its internal PTM context when any of the following occur:
 - A PTM Request is replayed.
 - A duplicate PTM ResponseD TLP is received.

- The relationship between PTM Master Time and the Upstream Port's local time changes, as determined by implementation specific criteria. For example, this may occur as a result of a transition to a non-D0 state or due to accumulated PPM drift.

These events are grouped under the label “Local Time Invalidation Event” in § Figure 6-24 .

- If ePTM is supported, an Upstream Port, upon replaying a PTM TLP, must invalidate its PTM context until two successive PTM dialogs have been completed successfully and without replays.

Base 6.4 vs Base 6.3

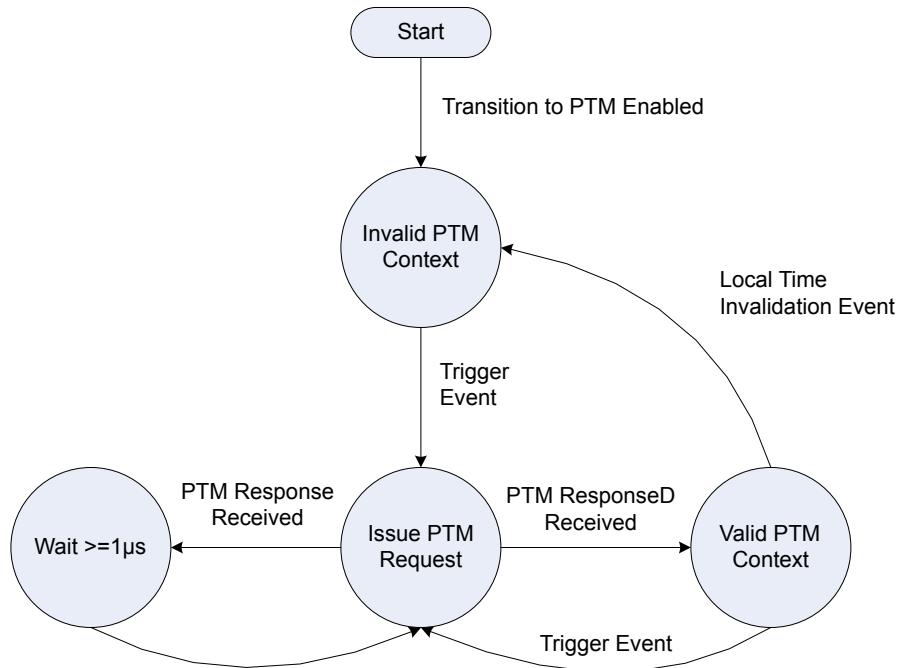


Figure 6-24 PTM Requester Operation §

IMPLEMENTATION NOTE: PTM INVALIDATION ON THE RECEPTION OF DUPLICATE TLPS §

Duplicate TLPs are detected and discarded in the Data Link Layer, whereas PTM messages are identified in the Transaction Layer. In some implementations it may be difficult or excessively complicated to distinguish a duplicate PTM TLP from other duplicate TLPs.

Because Upstream Ports are permitted to invalidate their internal PTM context for implementation specific criteria, a PTM Requester is allowed to invalidate its internal PTM context upon the reception of any duplicate TLP in addition to any duplicate PTM TLP. Similarly, if ePTM is supported, then a PTM Responder is allowed to invalidate its historical timestamps ($t_2 - t_3$) upon the reception of any duplicate TLP.

6.21.3.2 PTM Responder Role §

- Support for the PTM Responder role is indicated by setting the PTM Responder Capable bit in the PTM Capability register.
- Switches and Root Complexes are permitted to implement the PTM Responder Role.
 - A PTM capable Switch, when enabled for PTM by setting the PTM Enable bit in the PTM Control register associated with the Switch Upstream Port, must respond to all PTM Request Messages received at any of its Downstream Ports.
 - The mechanism by which Root Complexes communicate PTM Master Time to RCiEPs is implementation specific.
- PTM Responders must populate PTM ResponseD Messages as follows (refer to § Figure 6-22 and the accompanying implementation note):
 - The PTM Master Time field is a 64-bit value containing the value of PTM Master Time at the receipt of the PTM Request Message for the current PTM Dialog. In § Figure 6-22 , for the 2nd PTM dialog, this is the PTM Master Time at time t2'.
 - The Propagation Delay field is a 32-bit value containing the interval between the receipt of the PTM Request Message and the transmission of the PTM Response Message for the previous PTM dialog. In § Figure 6-22 , for the 2nd PTM dialog, this is the time interval between t2 and t3 captured during the 1st PTM dialog.
 - The unit of measurement for both fields is one ns.
 - A PTM Responder with multiple Downstream Ports must populate all PTM ResponseD Messages with values from a single PTM Root across all its Downstream Ports.
- Switch Downstream Ports and Root Ports acting as PTM Responders must respond to each PTM Request Message received at their Downstream Ports with either PTM Response or PTM ResponseD according to the following rules:
 - A PTM Responder must not send a PTM Response or PTM ResponseD Message without first receiving a PTM Request Message.
 - Upon receipt of a PTM Request Message, a PTM Responder must attempt to issue a PTM Response or PTM ResponseD Message within 10 µs.
 - A PTM Responder must issue PTM Response when the Downstream Port does not have valid historical timestamps (t3 - t2) with which to fulfill a PTM Request Message.
 - If ePTM is supported, a PTM Responder must invalidate its historical timestamps (t3 - t2) immediately upon replaying any PTM Response or PTM ResponseD.
 - If ePTM is supported a PTM Responder must, and if ePTM is not supported a PTM Responder is recommended to, invalidate its historical timestamps (t3 - t2) after receiving any duplicate PTM Request.
- A PTM Responder must issue PTM ResponseD when it has stored copies of the values required to populate the PTM ResponseD Message: historical timestamps (t3 - t2) and the PTM Master Time at the receipt of the most recent PTM Request Message (time t2').
- A PTM Responder is permitted to issue PTM Response when it has stored copies of the historical timestamps (t3 - t2) but must request the PTM Master Time from elsewhere. In this case, it is permitted to issue PTM Response messages in response to PTM Request Messages while it retrieves the PTM Master Time if that retrieval is expected to take more than 10 µs.

- The perceived granularity of the historical timestamps and PTM Master Time values transmitted by a PTM Responder must not exceed that reported in the Local Clock Granularity field of the PTM Capability Register.

6.21.3.3 PTM Time Source Role - Rules Specific to Switches §

In addition to the requirements listed above for the PTM Requester and PTM Responder Roles, Switches must follow these requirements:

- When the Upstream Port is associated with a Multi-Function Device, only a single Function associated with that Upstream Port is permitted to implement the PTM Extended Capability structure. For Switches, all PTM functionality associated with the Switch must be controlled through that structure. It is not required that the Function implementing the PTM Extended Capability structure be the Switch Upstream Port Function.
- The PTM Extended Capability structure for a Switch must indicate support for both the PTM Requester and PTM Responder roles.
- The PTM Extended Capability in the Upstream Port controls all Switches in that Upstream Port.
- A Switch is permitted to act as a PTM Root, or to issue PTM Requests on its Upstream Port to obtain the PTM Master Time for use in fulfilling PTM Requests received at its Downstream Ports. In the latter case the Switch must account for any internal delays within the Switch.
- A Switch is permitted to maintain a local PTM context for use in fulfilling PTM Requests received on its Downstream Ports.
- A Switch which is not acting as a PTM Root must invalidate its local context no more than 10 ms from the last PTM dialog on its Upstream Port. The Switch must then refresh its local PTM context prior to issuing further PTM ResponseD Messages on its Downstream Ports. This requirement for periodic refreshes is optional if it is guaranteed by implementation specific means that the Switch's local clock is phase locked with PTM Master Time.
- Any Switch implementing a local clock for the purpose of maintaining a local PTM context must report the granularity of this clock as defined in the PTM Capabilities structure (§ Section 7.9.15).

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: PTM TIMESTAMP CAPTURE MECHANISMS §

PTM uses services from both the Data Link and Transaction Layers. Accuracy requires that time measurements be taken as close to the Physical Layer as possible. Conversely, the messaging protocol itself properly belongs to the Transaction Layer. The PTM message protocol applies to a single Link, where the Upstream Port is the requester and the Downstream Port is the responder.

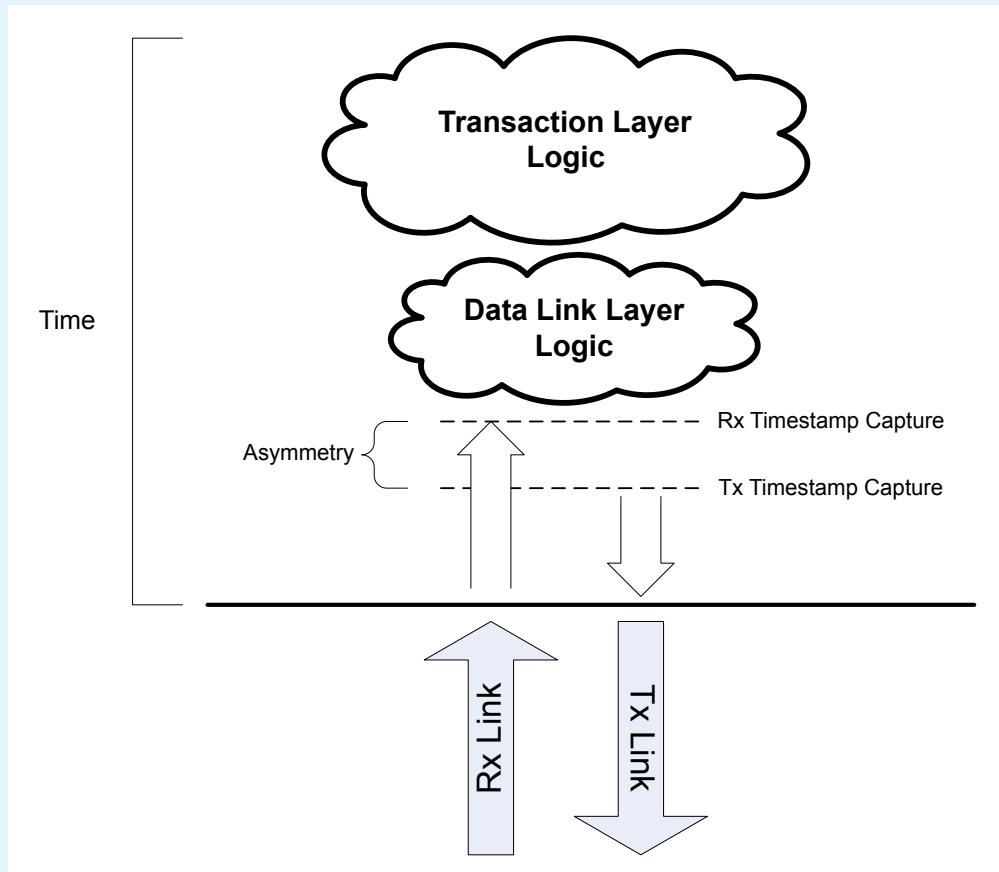


Figure 6-25 PTM Timestamp Capture Example §

§ Figure 6-25 illustrates how to select suitable timestamp capture points. For some implementations, the logic within the Transaction Layer and Data Link Layers is non-deterministic. Implementation details and current conditions have considerable impact on exactly when a particular packet may encounter any particular processing step. This makes it effectively impossible to capture any timestamp that accurately records the time of a particular physical event if timestamps are captured in the higher layers.

6.22 Readiness Notifications (RN) §

Readiness Notifications (RN) is intended to reduce the time software needs to wait before issuing Configuration Requests to a Device or Function following DRS Events or FRS Events. RN includes both the Device Readiness Status (DRS) and

Function Readiness Status (FRS) mechanisms. These mechanisms provide a direct indication of Configuration-Readiness (see Terms and Acronyms entry for Configuration-Ready). When used, DRS and FRS allow an improved behaviour over the Configuration RRS mechanism, and eliminate its associated periodic polling time of up to 1 second following a reset. With appropriate system support, the DRS mechanism can be used to support Devices that require more than 1 second to become Configuration-Ready (see HARDWARE/SOFTWARE RECOMMENDATIONS FOR OPTIMIZING CONFIGURATION READINESS below).

It is permitted that system software/firmware provide mechanisms that supersede the FRS and/or DRS mechanisms, however such software/firmware mechanisms are outside the scope of this specification.

IMPLEMENTATION NOTE: HARDWARE/SOFTWARE RECOMMENDATIONS FOR OPTIMIZING CONFIGURATION READINESS §

It is strongly recommended that implementers of System Software avoid unnecessary delays wherever possible. It is strongly recommended that hardware be designed to eliminate or minimize required delays, and utilize mechanisms provided in this and related specifications to communicate any required delays. Hardware implementers should document hardware behavior sufficiently to enable System Software to implement optimal behaviors.

Even before a Link is in L0, it is possible for System Software to determine if the DRS mechanism can be used to determine when a device becomes Configuration Ready. System Software may do this as follows:

1. If the DRS Supported bit in the Downstream Port above the device is Clear, stop this procedure and follow the procedure in § Section 2.3.1.
2. Check the Downstream Component Presence field in the Downstream Port.
 - o If Downstream Component Presence equals Link Up – Component Present and DRS Received, the device is already Configuration Ready.
 - o If System Software is informed, through implementation specific mechanisms, that the device supports DRS, continue with Step 3. System Software is strongly recommended to take advantage of such knowledge when available.
 - o If Downstream Component Presence equals Link Down – Flit Mode Negotiation Completed or Link Up Component Present:
 - If Flit Mode Status is Set, assume the device also supports DRS since DRS is mandatory if a component supports Flit Mode. Continue with Step 3.
 - Otherwise, continue with Step 7.
 - o If Otherwise, if the Link is Up, continue with Step 7.

When System Software determines that DRS is supported by both the Downstream Port and the device below it, the following DRS-Only procedure is strongly recommended:

3. The timeout based procedure defined in § Section 2.3.1 is not used. That procedure supports devices that can take a maximum of 1 second to become Configuration Ready. This DRS-Only procedure supports devices that take longer than 1 second to become Configuration Ready.
4. Either DRS Message Received or Downstream Component Presence are used to determine when the device is or becomes Configuration Ready.
5. The Downstream Component Presence field is used to avoid polling when a component is not present.
6. A non-blocking polling capability is implemented so that unrelated operations are not stalled. Software may configure DRS Signaling Control to generate an interrupt or FRS Message to indicate when polling should occur. It may be desirable to implement a timeout mechanism to terminate polling and indicate an error condition if the timeout expires. The timeout should be determined by system use case requirements. If none applies and a component is known to be present, a value of 10 seconds is recommended.

When System Software cannot determine that DRS is supported by both the Downstream Port and the device below it, then this hybrid approach for determining Configuration Ready is recommended:

7. The timeout based procedure defined in § Section 2.3.1 is run in parallel with the DRS-Only procedure in Step 4 through Step 6.
8. The receipt of a DRS Message indicates the Device is Configuration Ready, and System Software proceeds without further delay to configure the device.
9. If no DRS Message is received within an appropriate period determined by system use case requirements, then System Software may issue a Configuration Request to the Device per the procedure described in § Section 2.3.1.

6.22.1 Device Readiness Status (DRS) §

DRS MUST@FLIT be implemented, and the DRS Supported bit in the Link Capabilities 2 register MUST@FLIT be Set in all Downstream Ports and in Function 0 of all Upstream Ports. DRS is optional for Ports that do not support Flit Mode.

DRS must be used to indicate when a Device is Configuration-Ready following any of the following Device-level occurrences, which are subsequently referred to as “DRS Events”:

- Exit from Cold Reset
- Exit from Warm Reset, Hot Reset, Loopback, or Disabled
- Exit from L2/L3 Ready
- Any other scenario where the Port transitions from DL_Down to DL_Up status.

The DRS Message protocol requirements include the following:

- There is no enable or disable mechanism for DRS.
- It is expressly permitted for Upstream Ports to send DRS Messages even when the DRS Supported bit is Clear.
- A DRS Message must be transmitted by a DRS-capable Upstream Port following every DL_Down to DL_Up transition when all non-VF Functions on the Logical Bus associated with that Upstream Port become ready.
 - A Type 0 Function is ready when it is Configuration-Ready.
 - A Type 1 Function that is a Switch Upstream Port is ready when it is Configuration-Ready and all Functions on its secondary bus are Configuration-Ready.
 - A Type 1 Function that is not a Switch Upstream Port is ready when the Function itself is Configuration-Ready.
- After a Device transmits a DRS Message, non-VF Functions indicated as Configuration-Ready by that DRS Message must not return Completions with RRS in response to Configuration Requests unless a subsequent DRS Event occurs.

Additional requirements relating to Switches implementing DRS include:

- Must support DRS functionality in all Ports
- Implementation at each Downstream Port of the DRS Signaling Control field.
- For any physically-integrated Device that appears beneath a Switch Downstream Port, the DRS sent by the Switch does not indicate Configuration Readiness for that Device
 - For such a Device, implementation and use of DRS is independent of the Switch

Additional requirements for Root Ports and Switch Downstream Ports include:

Implementation of the DRS Message Received bit, which indicates receipt of a DRS Message

IMPLEMENTATION NOTE: DRS MESSAGES AND ACS SOURCE VALIDATION §

Functions are permitted to transmit DRS Messages before they have been assigned a Bus Number. Such messages will have a Requester ID with a Bus Number of 00h. If the Downstream Port has ACS Source Validation enabled, these Messages (see § [Section 6.12.1.1](#)) will likely be detected as an ACS Violation error.

6.22.2 Function Readiness Status (FRS) §

When implemented, FRS must be used to indicate a specific Function as being Configuration-Ready following any of the following Function-level occurrences, which are subsequently referred to as “FRS Events”:

- Function Level Reset (FLR)
- Completion of D3Hot to D0 transition
- Setting or Clearing of VF Enable in a PF (SR-IOV)

The FRS Message protocol requirements include the following:

- The Requester ID of the FRS Message must indicate the Function that has changed readiness status (see § [Section 2.2.8.6.3](#))
- The FRS Reason field in the FRS Message must indicate why that Function changed readiness status
- After a Function transmits an FRS Message, the indicated Function(s) must not return Completions with RRS in response to a Configuration Request unless a subsequent DRS Event or FRS Event occurs

Additional requirements for Switches implementing FRS include:

- Must support FRS functionality in the Upstream Port and all Downstream Ports
- The ability to transmit FRS Messages Upstream when required by the FRS protocol

Additional requirements for Physical Functions (PFs) include:

- The ability to transmit FRS Message Upstream when the VF Enable or VF Disable process completes

Additional requirements for Root Ports and Root Complex Event Collectors implementing FRS include:

- Must implement the FRS Queuing Extended Capability (see § [Section 7.8.10](#))

6.22.3 FRS Queuing §

Root Ports and Root Complex Event Collectors that support FRS must implement the FRS Queuing Extended Capability (see § [Section 7.8.10](#)).

For a Root Port, the FRS Message Queue contains FRS Messages received by the Root Port or generated by the Root Port.

For a Root Complex Event Collector , the FRS Message Queue contains FRS Messages generated by RCiEPs associated with the Root Complex Event Collector (see § Section 7.9.10) or generated by the Root Complex Event Collector .

The FRS Message Queue must satisfy the following requirements:

- The FRS Message Queue must be empty following Reset.
- For a Root Port, the FRS Message Queue must be emptied when the Link goes to DL_Down .
- FRS Messages must be queued in the order received.
- If the FRS Message Queue is not full at the time an FRS Message is received or is internally generated, that FRS Message must be entered in the queue and the FRS Message Received bit must set to 1b.
- If the FRS Message Queue is full at the time an FRS Message is received or is internally generated, that FRS Message must be discarded and the FRS Message Overflow bit must be set to 1b. The pre-existing FRS Message Queue must be preserved.
- The oldest FRS Message must be visible in the FRS Message Queue register (see § Section 7.8.10.4).
- Writing the FRS Message Queue register must remove the oldest element from the queue.
- When either FRS Message Received or FRS Message Overflow transitions from 0b to 1b, an interrupt must be generated if enabled.

6.23 Enhanced Allocation §

The Enhanced Allocation (EA) Capability is an optional Capability that allows the allocation of I/O, Memory and Bus Number resources in ways not possible with the BAR and Base/Limit mechanisms in the Type 0 and Type 1 Configuration Headers.

It is only permitted to apply EA to certain functions, based on the hierarchical structure of the functions as seen in PCI configuration space, and based on certain aspects of how functions exist within a platform environment (see § Figure 6-26).

Base 6.4 vs Base 6.3

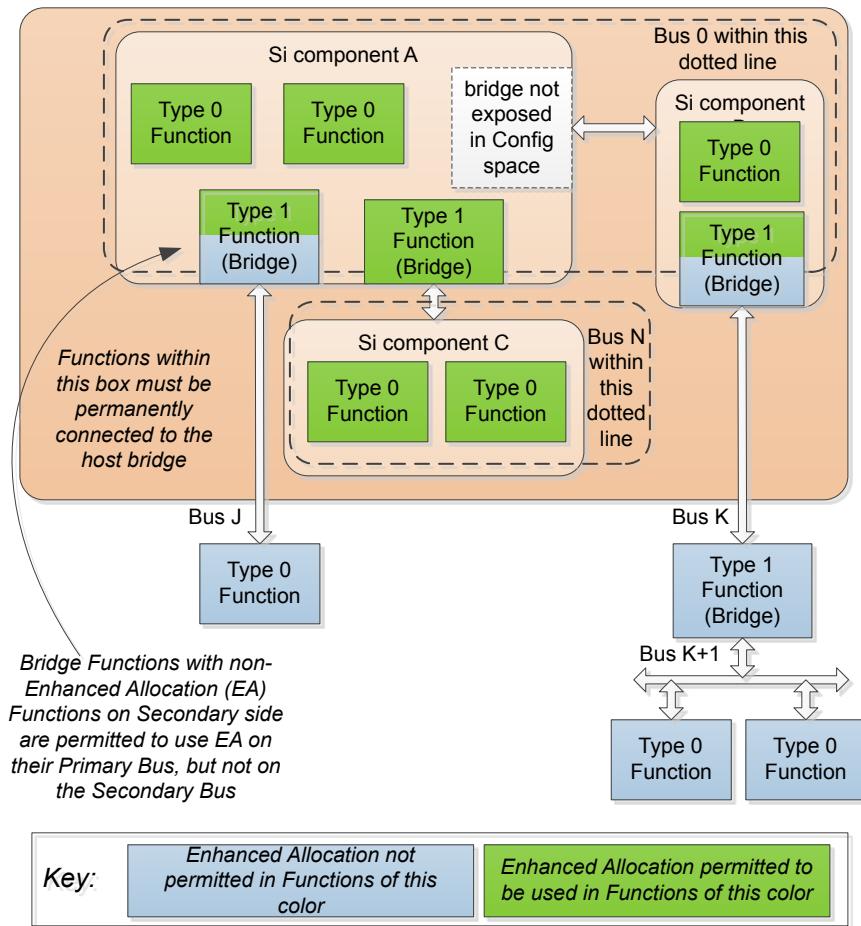


Figure 6-26 Example Illustrating Application of Enhanced Allocation §

Only functions that are permanently connected to the host bridge are permitted to use EA. A bridge function (i.e., any function with a Type 1 Configuration Header), is permitted to use EA for both its Primary Side and Secondary Side if and only if the function(s) behind the bridge are also permanently connected (below one or more bridges) to the host bridge, as shown for “Si component C” in § Figure 6-26 .

A bridge function is permitted to use EA only for its Primary Side if the function(s) behind the bridge are not permanently connected to the bridge, as with the bridges above Bus J and Bus K in § Figure 6-26 , and in this case the non-EA resource allocation mechanisms in the Type 1 Header for Bus numbers, MMIO ranges and I/O ranges are used for the Secondary side of the bridge. System software must ensure that the allocated Bus numbers are within the range indicated in the Fixed Secondary Bus Number and Fixed Subordinate Bus Number registers of the EA capability. System software must ensure that the allocated MMIO and I/O ranges are within ranges indicated with the corresponding Properties in the EA capability for resources to be allocated behind the bridge. For Bus numbers, MMIO and I/O ranges behind the bridge, hardware is permitted to indicate overlapping ranges in multiple bridge functions, however, in such cases, system software must ensure that the ranges actually assigned are non-overlapping.

Functions that rely exclusively on EA for I/O and Memory address allocation must hardwire all bits of all BARs in the PCI Header to 0. Such Functions must be clearly documented as relying on EA for correct operation, and platform integrators must ensure that only EA-aware firmware/software are used with such Functions.

When a Function allocates resources using EA and indicates that a resource range is associated with an equivalent BAR number, the Function must not request resources through the equivalent BAR, which must be indicated by hardwiring all bits of the equivalent BAR to 0.

For a bridge function that is permitted to implement EA based on the rules above, it is permitted, but not required, for the bridge function to use EA mechanisms to indicate resource ranges that are located behind the bridge Function. In the example shown in [§ Figure 6-26](#), the bridge above Bus N is permitted to use EA mechanisms to indicate the resources used by the two functions in “Si component C”, or that bridge is permitted to not indicate the resources used by the two functions in “Si component C”. System firmware/software must comprehend that such bridge functions are not required to indicate inclusively all resources behind the bridge, and as a result system firmware/software must make a complete search of all functions behind the bridge to comprehend the resources used by those functions.

A Function with an Expansion ROM is permitted to use the existing mechanism or the EA mechanism, but is not permitted to support both. If a Function uses the EA mechanism (EA entry with BEI of 8), the Expansion ROM Base Address and Expansion ROM Enable fields must be hardwired to 0 (see [§ Section 7.5.1.2.4](#)). The Enable bit of the EA entry is equivalent to the Expansion ROM Enable bit. If a Function uses Expansion ROM Base Address Register mechanism, no EA entry with a BEI of 8 is permitted. In both cases, Expansion ROM Validation, if supported, uses the Expansion ROM Validation Status and Expansion ROM Validation Details fields (see [§ Section 7.5.1.2.4](#)).

The requirements for enabling and/or disabling the decode of I/O and/or Memory ranges are unchanged by EA, including but not limited to the Memory Space and I/O Space enable bits in the Command register.

Any resource allocated using EA must not overlap with any other resource allocated using EA, except as permitted above for identifying permitted address ranges for resources behind a bridge.

6.24 Emergency Power Reduction State §

Emergency Power Reduction State is an optional mechanism to request that Functions quickly reduce their power consumption. Emergency Power Reduction is a fail-safe mechanism intended to be used to prevent system damage and is not intended to provide normal dynamic power management.

If a Function implements Emergency Power Reduction State, it must also implement the Power Budgeting extended capability and must report Power Budgeting values for this state (see [§ Section 7.8.1](#)). Devices that are integrated on the system board are not required to implement the Power Budgeting extended capability, but if they do so, they must meet the preceding requirement.

Functions enter and exit this state either autonomously or based on external requests. External requests may be either following a signaling protocol defined in an applicable form factor specification, or by a vendor-specific method. [§ Table 6-16](#) defines how the Emergency Power Reduction Supported and Emergency Power Reduction Initialization Required fields determine the mechanisms that are allowed to trigger entry and exit from this state (see [§ Section 7.5.3.15](#)).

Table 6-16 Emergency Power Reduction Supported Values §

| Emergency Power Reduction Supported | Emergency Power Reduction Initialization Required | Entry/Exit Permitted by | | |
|-------------------------------------|---|-------------------------|------------------------------|-----------------------|
| | | Form Factor Mechanism | Vendor Specific Mechanism(s) | Autonomous Mechanisms |
| 00b | 0 | No | Yes | Yes |
| | 1 | No | No | No |
| 01b | Any | No | Yes | Yes |
| 10b | Any | Yes | Yes | Yes |
| 11b | | Reserved | | |

Functions may indicate that they require re-initialization on exit from this state:

- If the Emergency Power Reduction Initialization Required bit is Clear (see § [Section 7.5.3.15](#)):
 - On entry to this state, the Function either operates normally (perhaps with reduced performance), or enters a device specific “power reduction dormant state”. The Upstream Port of the Device remains operating. Outstanding requests initiated by or directed to the Function must complete normally.
 - On exit from this state, the Function operates normally (perhaps resuming normal performance). Functions that entered a “power reduction dormant state” exit that state. In either case, no software intervention is required.
- If the Emergency Power Reduction Initialization Required bit is Set (see § [Section 7.5.3.15](#)):
 - On entry to this state, the Function ceases normal operation. The Upstream Port of the associated Device is permitted to enter [DL_Down](#).
 - If the Upstream Port remains in [DL_Up](#), outstanding requests directed to or initiated by the Function must complete normally.
 - If the Upstream Port enters [DL_Down](#), outstanding request behavior is defined in § [Section 2.9.1](#). This transition may result in a [Surprise Down](#) error.
 - Sticky bits must be preserved in this state.
 - On exit from this state, software intervention is required to resume normal operation. The mechanism used to indicate to software when this is required is outside the scope of this specification (e.g., a device specific interrupt). If the Upstream Port entered [DL_Down](#), all Functions of the Device are reset and a full reconfiguration is required (see § [Section 2.9.2](#)).

The following rules apply to the Emergency Power Reduction State:

- A Device supports Emergency Power Reduction State if at least one Function in the Upstream Port indicates support (i.e., Emergency Power Reduction Supported is non-zero).
- Emergency Power Reduction State is associated with a Device. All Functions in a Device that support it enter and exit this state at the same time.
- For ~~↓↓SR-IOV devices, ↓↑SR-IOV Devices, ↑~~ if the Emergency Power Reduction Supported field in a VF is non-zero, that VF enters and exits the Emergency Power Reduction State at the same time as its associated PF. For such VFs, the Emergency Power Reduction Detected bit must be hardwired to Zero, but software can use the associated PF's bit to emulate the bit in its VFs.
- Functions where the Emergency Power Reduction Supported field is 00b are not affected by the Emergency Power Reduction State of the Device as long as the Upstream Port remains in [DL_Up](#). The Emergency Power Reduction Detected bit is [RsvdZ](#).
- Functions where the Emergency Power Reduction Supported field is 01b or 10b:
 - Set the Emergency Power Reduction Detected bit when the Device enters Emergency Power Reduction State.
 - Clear the Emergency Power Reduction Detected bit when requested if the Device has exited the Emergency Power Reduction State.
- For Switches, Downstream Switch Ports enter and exit Emergency Power Reduction State at the same time as the associated Upstream Switch Port. The corresponding fields in Configuration Space are reserved for Downstream Switch Ports.
- For SR-IOV Devices, VFs enter and exit Emergency Power Reduction State at the same time as their PF. The corresponding fields in Configuration Space are reserved for VFs.
- Encoding 10b shall not be used unless the associated form factor specification defines a mechanism for requesting Emergency Power Reduction.

- It is strongly recommended that the Emergency Power Reduction Supported field be initialized by hardware or firmware within the Function prior to initial device enumeration. This initialization is permitted to be deferred to device driver load when this is not practical (e.g., when there is no firmware ROM).

IMPLEMENTATION NOTE: DIAGNOSTIC CHECKING OF EMERGENCY POWER REDUCTION DETECTED §

The Emergency Power Reduction Detected bit permits system software to detect that Emergency Power Reduction State was entered, even momentarily. The Emergency Power Reduction Request bit can be used by software to request entry. Normally, software would use a system specific method to enter the Emergency Power Reduction State using external mechanisms.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: EMERGENCY POWER REDUCTION STATE: EXAMPLE ADD-IN CARD §

§ Figure 6-27 shows an example multi-Device add-in card supporting Emergency Power Reduction. Note that Device C does not support the Emergency Power Reduction State. Device C might be a Switch that fans out to Devices A and B.

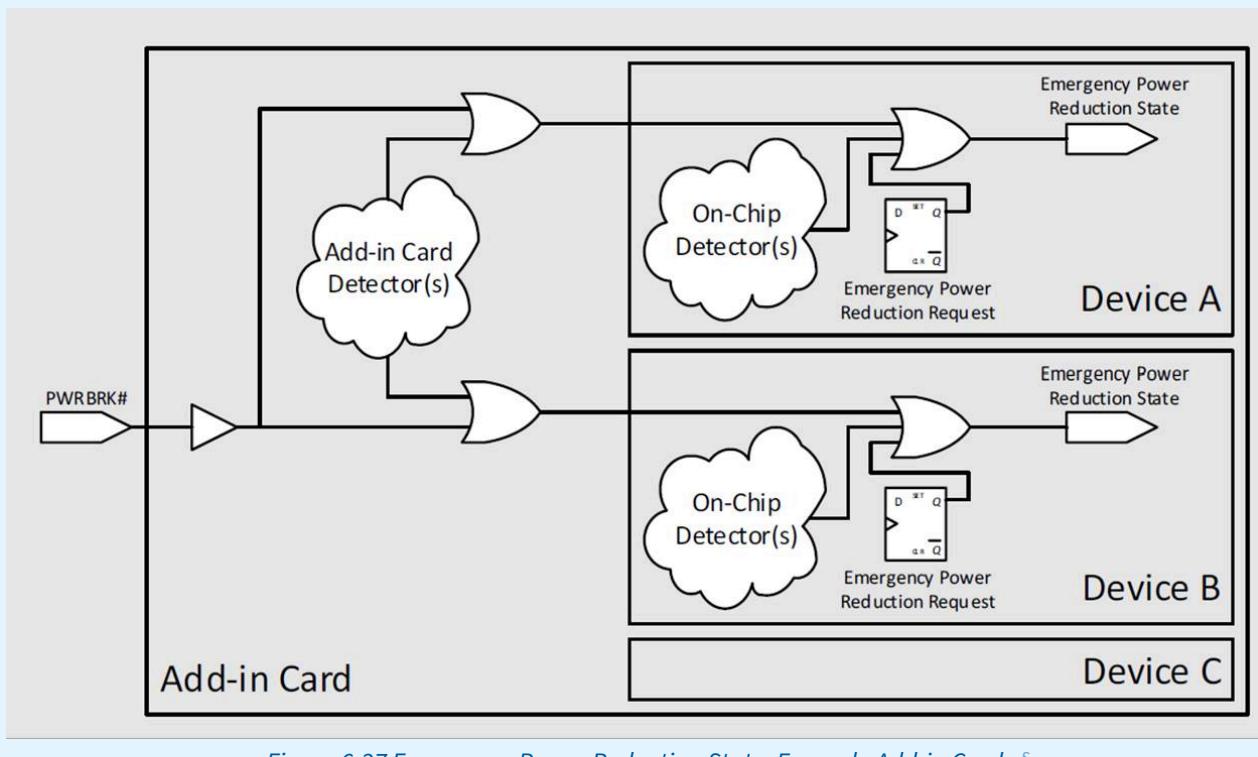


Figure 6-27 Emergency Power Reduction State: Example Add-in Card §

6.25 Hierarchy ID Message §

When software initializes a PCI Hierarchy, it assigns unique Bus and Device numbers to each component so that every Function in the Hierarchy has a unique Routing ID within that Hierarchy. To ensure that Routing IDs are unique in large systems that contain more than one Hierarchy and in clustered systems that contain multiple Hierarchies, additional information is required to augment the Routing ID to produce a unique number. Functions can be uniquely identified by the combination of:

- Unique Identifier for the System (or Root Complex)
- Unique Identifier for the Hierarchy within that Root Complex
- Routing ID within that Hierarchy

The Hierarchy ID Message (see § [Section 2.2.8.6.4](#)) is used to provide the additional information needed for a Function to uniquely identify itself in a multi-hierarchy platform.

Hierarchy ID Messages are generated by a Downstream Port upon software request. Received messages at an Upstream Port are reported in the Hierarchy ID Extended Capability (see § [Section 7.9.17](#)).

Hierarchy ID Messages are a PCI-SIG-Defined Type 1 VDM. Hierarchy ID Messages can safely be sent at any time and components that do not comprehend them will silently ignore them.

Hierarchy ID Messages typically are sent from a Downstream Port at the top of the Hierarchy (e.g., a Root Port). In systems where the Root Port does not support Hierarchy ID Messages, Hierarchy ID Messages can be sent from Switch Downstream Ports.

The Hierarchy ID Message is intended for use by software, firmware, and/or hardware. When using the Hierarchy ID Message, all bits of the Hierarchy ID, System GUID, System GUID Authority ID fields must be compared, without regard to any internal structure. How this information is used is outside the scope of this specification.

Layout of the Hierarchy ID Message is shown in [Figure 2-61](#). Fields in the Hierarchy ID Message are as follows:

Hierarchy ID contains the Segment Group Number associated with this Hierarchy (as defined by the *PCI Firmware Specification*). This field can be used in conjunction with the Routing ID to uniquely identify a Function within a System. The value 0000h indicates the default (or only) Hierarchy of the Root Complex. Non-zero values indicate additional Hierarchies.

System GUID [143:0], in conjunction with System GUID Authority ID, provides a globally unique identification for a System.

System GUID[143:136] is byte 14 in the Hierarchy ID Message.

System GUID[135:128] is byte 15 in the Hierarchy ID Message.

System GUID[127:120] is byte 16 in the Hierarchy ID Message.

System GUID[119:112] is byte 17 in the Hierarchy ID Message.

System GUID[111:104] is byte 18 in the Hierarchy ID Message.

System GUID[103:96] is byte 19 in the Hierarchy ID Message.

System GUID[95:88] is byte 20 in the Hierarchy ID Message.

System GUID[87:80] is byte 21 in the Hierarchy ID Message.

System GUID[79:72] is byte 22 in the Hierarchy ID Message.

System GUID[71:64] is byte 23 in the Hierarchy ID Message.

System GUID[63:56] is byte 24 in the Hierarchy ID Message.

System GUID[55:48] is byte 25 in the Hierarchy ID Message.

System GUID[47:40] is byte 26 in the Hierarchy ID Message.

System GUID[39:32] is byte 27 in the Hierarchy ID Message.

System GUID[31:24] is byte 28 in the Hierarchy ID Message.

System GUID[23:16] is byte 29 in the Hierarchy ID Message.

System GUID[15:8] is byte 30 in the Hierarchy ID Message.

System GUID[7:0] is byte 31 in the Hierarchy ID Message.

System GUID Authority ID identifies the mechanism used to ensure that the System GUID is globally unique. The mechanism for choosing which Authority ID to use for a given system is implementation specific. The defined values are shown in § [Table 6-17](#).

Table 6-17 System GUID Authority ID Encoding §

| Authority ID | Description |
|--------------|---|
| 00h | None - System GUID[143:0] is not meaningful. |

| Authority ID | Description |
|--------------|---|
| | System GUID[143:0] must be 0. |
| 01h | <p>Timestamp - System GUID[63:0] contains a timestamp associated with the particular system. Encoding is a Unix 64 bit time (number of seconds since midnight UTC January 1, 1970).</p> <p>The mechanism of choosing the timestamp to represent a system is implementation specific.</p> <p>System GUID[143:64] must be 0.</p> |
| 02h | <p>IEEE EUI-48 - System GUID[47:0] contains a 48 bit Extended Unique Identifier (EUI-48) associated with the particular system. Encoding is defined by the IEEE. See [EUI-48] for details. EUI-48 values are frequently used as network interface MAC addresses.</p> <p>The mechanism of choosing the EUI-48 value to represent a system is implementation specific.</p> <p>System GUID[143:48] must be 0.</p> |
| 03h | <p>IEEE EUI-64 - System GUID[63:0] contains a 64 bit Extended Unique Identifier (EUI-64) associated with the particular system. Encoding is defined by the IEEE. See [EUI-64] for details.</p> <p>The mechanism of choosing the EUI-64 value to represent a system is implementation specific.</p> <p>System GUID[143:64] must be 0.</p> |
| 04h | <p>RFC-4122 UUID - System GUID[127:0] contain a UUID as defined by the IETF in [RFC-4122]. This definition is technically equivalent to [ITU-T-Rec-X-667] or [ISO-IEC-9834-8].</p> <p>The mechanism of choosing the UUID value to represent a system is implementation specific.</p> <p>System GUID[143:128] must be 0</p> |
| 05h | <p>IPv6 Address - System GUID[127:0] contains the unique IPv6 address of one of the network interfaces of the system.</p> <p>The mechanism of choosing the IPv6 value to represent a system is implementation specific.</p> <p>System GUID[143:128] must be 0.</p> |
| 06h to 7Fh | <p>Reserved - System GUID[143:0] contains a unique value. The mechanism used to ensure uniqueness is outside the scope of this specification.</p> |
| 80h to FFh | <p>PCI-SIG Vendor Specific - System GUID Authority ID values 80h to FFh are reserved for PCI-SIG vendor-specific usage.</p> <p>System GUID[143:128] contains a PCI-SIG assigned Vendor ID.</p> <p>System GUID[127:0] contain a unique number assigned by that vendor. The mechanism used for assigning numbers is implementation specific. One possible mechanism would be to use the serial number assigned to the system.</p> <p>The mechanism used to choose between these System GUID Authority IDs is implementation specific. One usage would be to allow a vendor to define up to 128 distinct 128-bit System GUID schemes.</p> |

IMPLEMENTATION NOTE: SYSTEM GUID CONSISTENCY AND STABILITY §

To support the purpose of System GUID, software should ensure that a single system uses identical System GUID and System GUID Authority ID values everywhere.

Implementers should carefully consider their stability requirements for the System GUID value. For example, some use cases may require that the value not change when the system is rebooted. In those cases, a mechanism that picks the EUI-48 value associated with the first Ethernet MAC address discovered might be problematic if the result changes due to hardware failure, system reconfiguration, or variations/parallelism in the discovery algorithm.

IMPLEMENTATION NOTE: HIERARCHY ID VS. DEVICE SERIAL NUMBER §

The Device Serial Number mechanism can also be used to uniquely identify a component (see § Section 7.9.3). Device Serial Number may be a more expensive solution to this problem if it involves a ROM associated with each component.

IMPLEMENTATION NOTE: VIRTUAL FUNCTIONS AND HIERARCHY ID §

The Hierarchy ID capability can be emulated by the Virtualization Intermediary (VI). Doing so provides VF software access to this Hierarchy ID information.

When VF hardware needs access to this information, the VF should implement the Hierarchy ID capability. This provides access to both VF software and hardware.

In some situations, the VF should get the same information as the PF. In other situations, particularly those involving migration of Virtual Machines, it may be appropriate to present the VF with Hierarchy ID information that differs from the associated PF and from other VFs associated with that PF.

The following mechanisms are supported:

| | VF Hierarchy ID Capability | Hierarchy ID VF Configurable | Hierarchy ID Writeable | VF Software has access | VF Hardware has access | VF Hierarchy Data / GUID |
|---|----------------------------|------------------------------|------------------------|------------------------|------------------------|--------------------------|
| 1 | Not Present | n/a | n/a | No | No | Not Emulated |
| 2 | | | | Yes | No | Emulated |
| 3 | Present | 0b | 0b | Yes | Yes | Same as PF |
| 4 | | 1b | 0b | Yes | Yes | Same as PF |
| 5 | | 1b | 1b | Yes | Yes | Configured by VI |

In mechanism 1, the Virtualization Intermediary does not emulate the capability. VF software and hardware have no access.

In mechanism 2, the Virtualization Intermediary emulates the capability and returns whatever Hierarchy ID information is desired. VF software has access. VF hardware does not have access.

In mechanisms 3 and 4, VF information is the same as the PF and is automatically filled in from received Hierarchy ID messages. Both VF hardware and software have access.

In mechanism 5, VF information is configured by software (probably the VI). Both VF hardware and software have access.

6.26 Flattening Portal Bridge (FPB) §

6.26.1 Introduction §

The Flattening Portal Bridge (FPB) is an optional mechanism which can be used to improve the scalability and runtime reallocation of Routing IDs and Memory Space resources.

For non-ARI Functions associated with an Upstream Port, the Routing ID consists of a 3-bit Function Number portion, which is determined by the construction of the Upstream Port hardware, and a 13-bit Bus Number and Device number portion, determined by the Downstream Port above the Upstream port.

For ARI Functions associated with an Upstream Port, the Routing ID consists of an 8-bit Function Number portion, and only the 8-bit Bus Number portion is determined by the Downstream Port above the Upstream port.

A bridge that implements the FPB Capability can itself also be referred to as an FPB. The FPB Capability can be applied to any logical bridge, as illustrated in § Figure 6-28 .

Base 6.4 vs Base 6.3

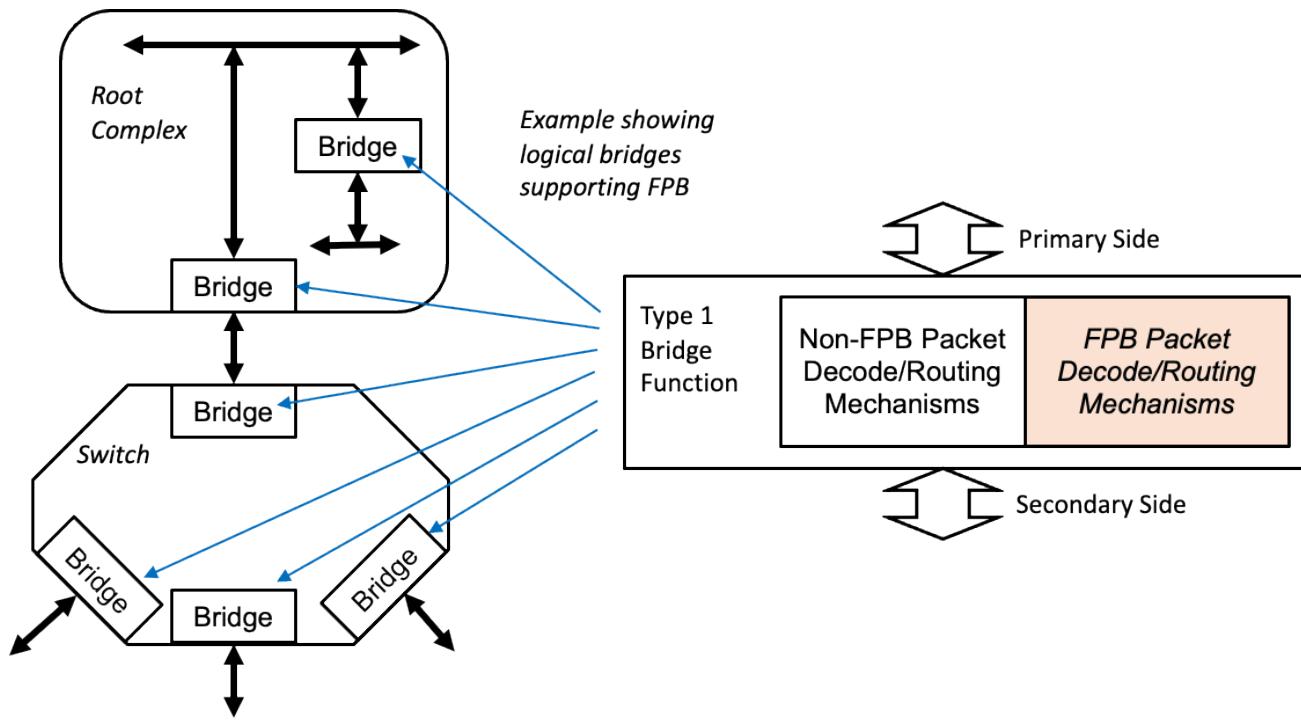


Figure 6-28 FPB High Level Diagram and Example Topology §

FPB changes the way Bus Numbers are consumed by Switches to reduce waste, by “flattening” the way Bus Numbers are used inside of Switches and by Downstream Ports (see § Figure 6-29).

Base 6.4 vs Base 6.3

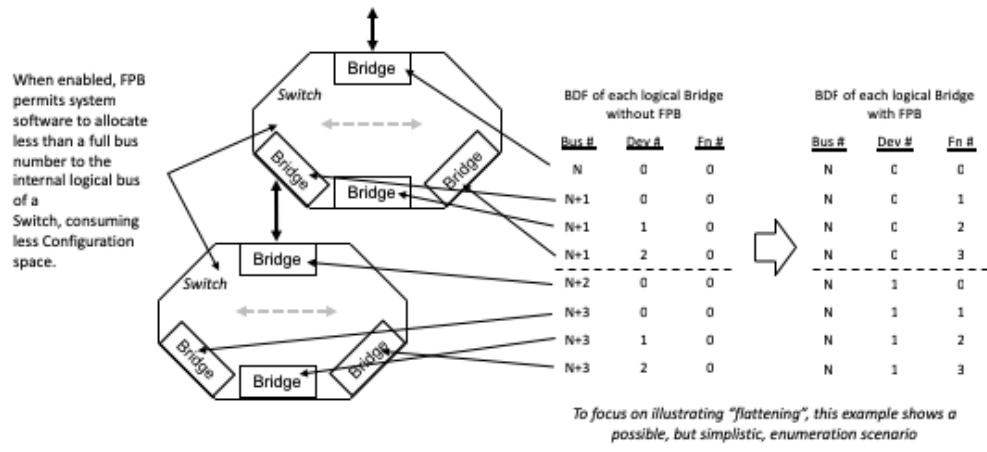


Figure 6-29 Example Illustrating “Flattening” of a Switch §

FPB defines mechanisms for system software to allocate Routing IDs and Memory Space resources in non-contiguous ranges, enabling system software to assign pools of these resources from which it can allocate “bins” to Functions below the FPB. This is done using a bit vector where each bit when Set assigns a corresponding range of resources to the Secondary Side of the bridge (see § Figure 6-30).

Base 6.4 vs Base 6.3

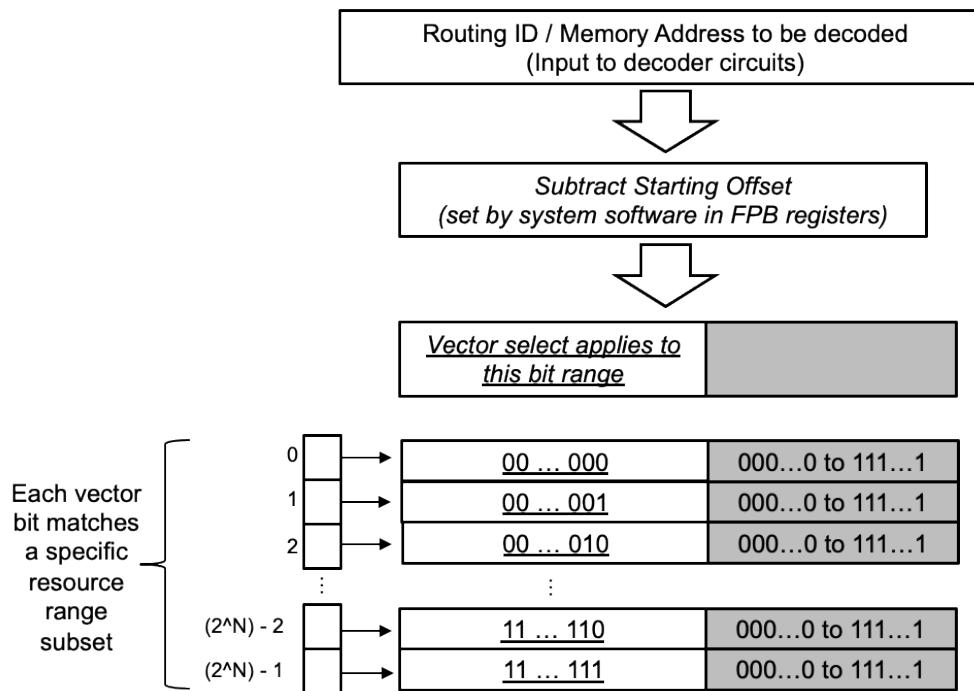


Figure 6-30 Vector Mechanism for Address Range Decoding §

This allows system software to assign Routing IDs and/or Memory Space resources required by a device hot-add without having to rebalance other, already assigned resource ranges, and to return to the pool resources freed, for example by a hot remove event.

FPB is defined to allow both the non-FPB and FPB mechanisms to operate simultaneously, such that, for example, it is possible for system firmware/software to implement a policy where the non-FPB mechanisms continue to be used in parts of the system where the FPB mechanisms are not required (see § Figure 6-31). In this figure, the decode logic is assumed to provide a 1b output when a given TLP is decoded as being associated with the bridge's Secondary Side. The non-FPB decode mechanisms apply as without FPB, so for example only the Bus Number portion (bits 15:8) of a Routing ID is tested by the non-FPB decode logic when evaluating an ID routed TLP.

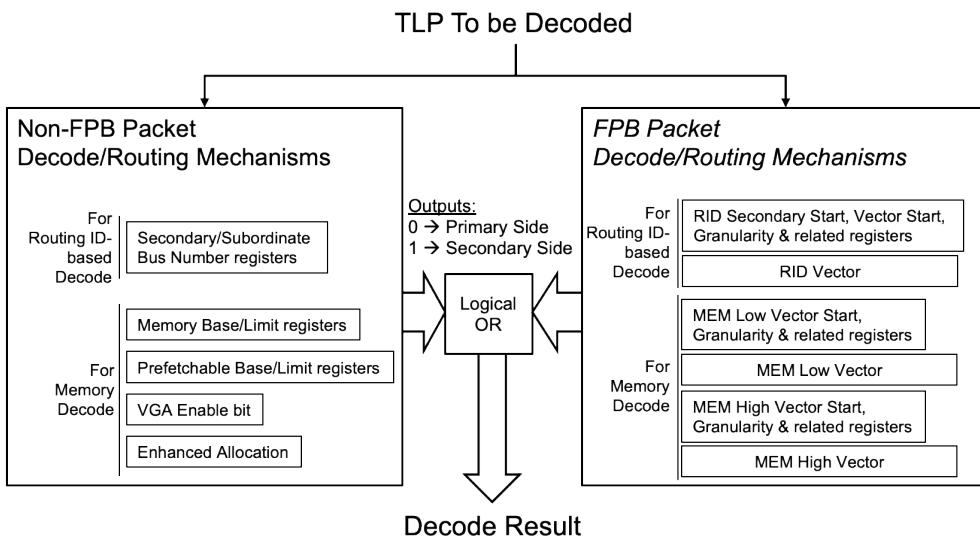


Figure 6-31 Relationship between FPB and non-FPB Decode Mechanisms §

It is important to recognize that, although FPB adds additional ways for a specific bridge to decode a given TLP, FPB does not change anything about the fundamental ways that bridges operate within the Switch and Root Complex architectural structures. FPB uses the same architectural concepts to provide management mechanisms for three different resource types:

1. Routing IDs
2. Memory below 4 GB (“MEM Low”)
3. Memory above 4 GB (“MEM High”)

A hardware implementation of FPB is permitted to support any combination of these three mechanisms. For each mechanism, FPB uses a bit-vector to indicate, for a specific subset range of the selected resource type, if resources within that range are associated with the Primary or Secondary side of the FPB. Hardware implementations are permitted to implement a small range of sizes for these vectors, and system firmware/software is enabled to make the most effective use of the available vector by selecting an initial offset at which the vector is applied, and a granularity for the individual bits within the vector to indicate the size of the resource range to which the bits in a given vector apply.

6.26.2 Hardware and Software Requirements §

The following rules apply when any of the FPB mechanisms are used:

- If system software violates any of the rules concerning FPB, the hardware behavior is undefined.
- It is permitted to implement FPB in any PCI bridge (Type 1) Function, and every Function that implements FPB must implement the FPB Capability (see § Section 7.8.11).
- If a Switch implements FPB then the Upstream Port and all Downstream Ports of the Switch must implement FPB.
- Software is permitted to enable FPB at some Switch Ports and not others.
- A Root Complex is permitted to implement FPB on some Root Ports but not on others.

- A Type 1 Function is permitted to implement the FPB mechanisms applying to any one, two or three of these elemental mechanisms:
 - Routing IDs (RID)
 - Memory below 4 GB (“MEM Low”)
 - Memory above 4 GB (“MEM High”)
- System software is permitted to enable any combination (including all or none) of the elemental mechanisms supported by a specific FPB.
- The error handling and reporting mechanisms, except where explicitly modified in this section, are unaffected by FPB.
- Following any reset of the FPB Function, the FPB hardware must Clear all bits in all implemented vectors.
- Once enabled (through the FPB RID Decode Mechanism Enable, FPB MEM Low Decode Mechanism Enable, and/or FPB MEM High Decode Mechanism Enable bits), if system software subsequently disables an FPB mechanism, the values of the entries in the associated vector are undefined, and if system software subsequently re-enables that FPB mechanism the FPB hardware must Clear all bits in the associated vector.
- If an FPB is implemented with the No_Soft_Reset bit Clear, when that FPB is cycled through D0 → D3_{Hot} → D0, then all FPB mechanisms must be disabled, and the FPB must Clear all bits in all implemented vectors.
- If an FPB is implemented with the No_Soft_Reset bit Set, when that FPB is cycled through D0 → D3_{Hot} → D0, then all FPB configuration state must not change, and the entries in the FPB vectors must be retained by hardware.
- Hardware is not required to perform any type of bounds checking on FPB calculations, and system software must ensure that the FPB parameters are correctly programmed
 - It is explicitly permitted for system software to program Vector Start values that cause the higher order bits of the corresponding vector to surpass the resource range associated with a given FPB, but in these cases system software must ensure that those higher order bits of the vector are Clear.
 - Examples of errors that system software must avoid include duplication of resource allocation, combinations of start offsets with set vector bits that could create “wrap-around” or bounds errors

The following rules apply to the FPB Routing ID (RID) mechanism:

- FPB hardware must consider a specific range of RIDs to be associated with the Secondary side of the FPB if the Bus Number portion falls within the Bus Number range indicated by the values programmed in the Secondary and Subordinate Bus Number registers logically OR'd with the value programmed into the corresponding entry in the FPB RID Vector.
- System software must configure the Configuration Request Type 1 to Type 0 conversion mechanisms in a Bridge Function before attempting to pass Configuration Requests through that Bridge.
- System software must either program the legacy and FPB mechanisms for Configuration Request Type 1 to Type 0 conversion in a Bridge Function such that they give identical results, or such that one of the two mechanisms is disabled.
 - If it is intended to use only the FPB RID mechanism for BDF decoding, then system software must ensure that both the Secondary and Subordinate Bus Number registers are 0.
 - If it is intended to enable the FPB RID Decode Mechanism, but to use only the legacy mechanism for Configuration Request Type 1 to Type 0 conversion, then system software must write bits 7:3 of the RID Secondary Start field to 0 0000b.
- System software must ensure that the FPB routing mechanisms are configured such that Configuration Requests targeting Functions Secondary side of the FPB will be routed by the FPB from the Primary to Secondary side of the FPB.

When ARI is not enabled, the FPB RID mechanism can be applied with different granularities, programmable by system software through the FPB RID Vector Granularity field in the FPB RID Vector Control 1 Register . § Figure 6-32 illustrates the relationships between the layout of RIDs and the supported granularities. The reader may find it helpful to refer to this figure when considering the requirements defined below and in the definition of the Flattening Portal Bridge (FPB) Capability (see § Section 7.8.6).

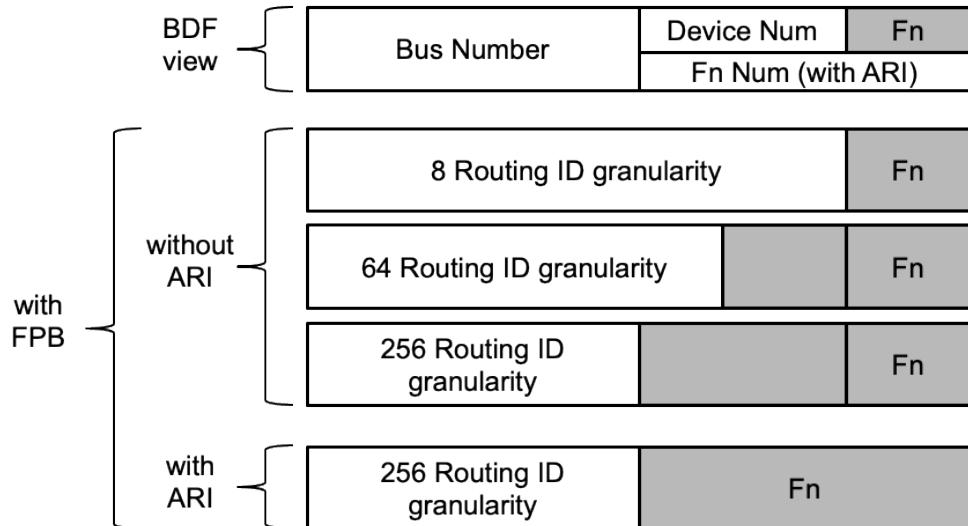


Figure 6-32 Routing IDs (RIDs) and Supported Granularities §

- System software must program the FPB RID Vector Granularity and FPB RID Vector Start fields in the FPB RID Vector Control 1 Register per the constraints described in the descriptions of those fields.
- For all FPBs other than those associated with Upstream Ports of Switches:
 - When ARI Forwarding is not supported, or when the ARI Forwarding Enable bit in the Device Control 2 Register is Clear, FPB hardware must convert a Type 1 Configuration Request received on the Primary side of the FPB to a Type 0 Configuration Request on the Secondary side of the FPB when bits 15:3 of the Routing ID of the Type 1 Configuration Request matches the value in the RID Secondary Start field in the FPB RID Vector Control 2 Register , and system software must configure the FPB accordingly.
 - When the ARI Forwarding Enable bit in the Device Control 2 Register is Set, FPB hardware must convert a Type 1 Configuration Request received on the Primary side of the FPB to a Type 0 Configuration Request on the Secondary side of the FPB when the Bus Number portion of the Routing ID of the Type 1 Configuration Request matches the value in the Bus Number address (bits 15:8 only) of the RID Secondary Start field in the FPB RID Vector Control 2 Register , and system software must configure the FPB accordingly.
- For FPBs associated with Upstream Ports of Switches only, when the FPB RID Decode Mechanism Enable bit is Set, FPB hardware must use the FPB Num Sec Dev field of the FPB Capabilities register to indicate the quantity of Device Numbers associated with the Secondary Side of the Upstream Port bridge, which must be used by the FPB in addition to the RID Secondary Start field in the FPB RID Vector Control 2 Register to determine when a Configuration Request received on the Primary side of the FPB targets one of the Downstream Ports of the Switch, determining in effect when such a Request must be converted from a Type 1 Configuration Request to a Type 0 Configuration Request, and system software must configure the FPB appropriately.
 - System software configuring FPB must comprehend that the logical internal structure of a Switch will change depending on the value of the FPB RID Decode Mechanism Enable bit in the Upstream Port of a Switch.

- Downstream Ports must use their corresponding RID values, and their Requester IDs and Completer IDs, as determined by the Upstream Port’s FPB Num Sec Dev and RID Secondary Start values
- All implemented Functions in the range determined by the Switch Upstream Port Function’s RID Secondary Start and FPB Num Sec Dev must be Switch Downstream Ports associated with that Switch Upstream Port; System Software is required to scan all Functions in this range to determine which are implemented.
- It is strongly recommended that System Software assign the RID Secondary Start such that the Bus and Device Numbers are not the same as for the Switch Upstream Port; otherwise, the resulting hardware behavior is undefined.
- For FPBs associated with Upstream Ports of Switches only, hardware must comprehend that Configuration Requests targeting the Upstream Port itself and any Downstream Ports of the Switch flattened into the range of Function Numbers with the same Bus and Device Numbers as the Upstream Port itself will be converted from Type 1 to Type 0 by the Downstream Port above the Switch, but any other Downstream Ports of the Switch flattened into successive Device Numbers will not be converted from Type1 to Type0 by the Downstream Port above the Switch and so must effectively be converted from Type 1 to Type 0 by the Switch Upstream Port itself.
This is a special case, but the concept is not unique to FPB, and is a reflection of the definition of the relationship between Bus/Device Numbers and Function Numbers – Function Numbers are always determined by the hardware of the Upstream Port, whereas the Bus and Device Numbers for an Upstream Port are always determined by the Downstream Port immediately above the Upstream Port.
- FPBs must implement bridge mapping for INTx virtual wires (see § Section 2.2.8.1)
- Hardware and software must apply this algorithm (or the logical equivalent) to determine which entry in the FPB RID Vector applies to a given Routing ID (RID) address:
 - IF the RID is below the value of FPB RID Vector Start , then the RID is out of range (below the start) and so cannot be associated with the Secondary side of the bridge, ELSE
 - calculate the offset within the vector by first subtracting the value of FPB RID Vector Start , then dividing this according to the value of FPB RID Vector Granularity to determine the bit index within the vector.
 - IF the bit index value is greater than the length indicated by FPB RID Vector Size Supported , then the RID is out of range (beyond the top of the range covered by the vector) and so cannot be associated with the Secondary side of the bridge, ELSE
 - if the bit value within the vector at the calculated bit index location is 1b, THEN the RID address is associated with the Secondary side of the bridge, ELSE the RID address is associated with the Primary side of the bridge.

The following rules apply to the FPB MEM Low mechanism:

The FPB MEM Low mechanism can be applied with different granularities, programmable by system software through the FPB MEM Low Vector Granularity field in the FPB MEM Low Vector Control Register . § Figure 6-33 illustrates the relationships between the layout of addresses in the memory address space below 4 GB to which the FPB MEM Low mechanism applies. The reader may find it helpful to refer to this figure when considering the requirements defined below and in the definition of the Flattening Portal Bridge (FPB) Capability (see § Section 7.8.11).

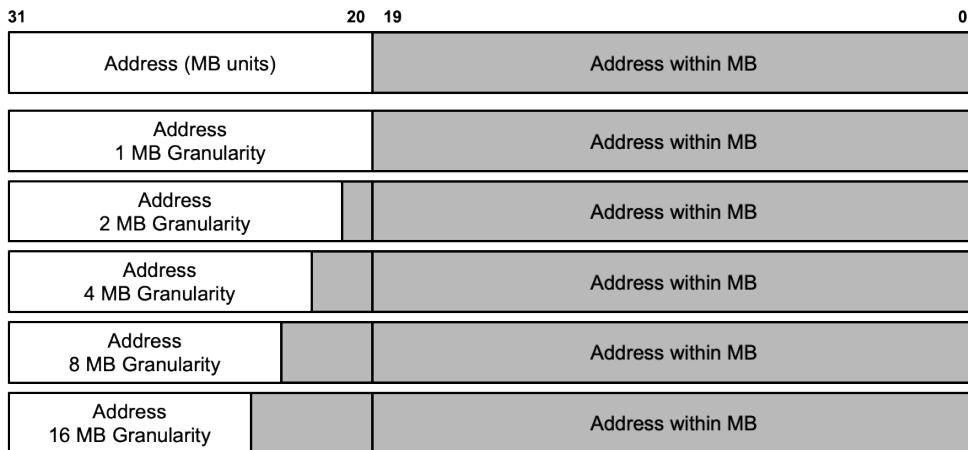


Figure 6-33 Addresses in Memory Below 4 GB and Effect of Granularity

- System software must program the FPB MEM Low Vector Granularity and FPB MEM Low Vector Start fields in the FPB MEM Low Vector Control Register per the constraints described in the descriptions of those fields.
- FPB hardware must consider a specific Memory address to be associated with the Secondary side of the FPB if that Memory address falls within any of the ranges indicated by the values programmed in other bridge Memory decode registers (enumerated below) logically OR'd with the value programmed into the corresponding entry in the FPB MEM Low Vector. Other bridge Memory decode registers include:
 - Memory Base/Limit registers
 - 64-bit Memory Base/Limit registers
 - VGA Enable bit in the Bridge Control Register
 - Enhanced Allocation (EA) Capability (if supported)
 - FPB MEM High mechanism (if supported and enabled)
- Hardware and software must apply this algorithm (or the logical equivalent) to determine which entry in the FPB MEM Low Vector applies to a given Memory address:
 - If the Memory address is below the value of FPB MEM Low Vector Start, then the Memory address is out of range (below) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
 - calculate the offset within the vector by first subtracting the value of FPB MEM Low Vector Start, then dividing this according to the value of FPB MEM Low Vector Granularity to determine the bit index within the vector.
 - If the bit index value is greater than the length indicated by FPB MEM Low Vector Size Supported, then the Memory address is out of range (above) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
 - if the bit value within the vector at the calculated bit index location is 1b, then the Memory address is associated with the Secondary side of the bridge, else the Memory address is associated with the Primary side of the bridge.

The following rules apply to the FPB MEM High mechanism:

- System software must program the FPB MEM High Vector Granularity and FPB MEM High Vector Start Lower fields in the FPB MEM High Vector Control 1 Register per the constraints described in the descriptions of those fields.

- FPB hardware must consider a specific Memory address to be associated with the Secondary side of the FPB if that Memory address falls within any of the ranges indicated by the values programmed in other bridge Memory decode registers (enumerated below) logically OR'd with the value programmed into the corresponding entry in the FPB MEM High Vector. Other bridge Memory decode registers include:
 - Memory Base/Limit registers
 - 64-bit Memory Base/Limit registers
 - VGA Enable bit in the Bridge Control Register
 - Enhanced Allocation (EA) Capability (if supported)
 - FPB MEM Low mechanism (if supported and enabled)
- Hardware and software must apply this algorithm to determine which entry in the FPB MEM High Vector applies to a given Memory address:
 - If the Memory address is below the value of FPB MEM High Vector Start Upper / FPB MEM High Vector Start Lower, then the Memory address is out of range (below) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
 - calculate the offset within the vector by first subtracting the value of FPB MEM High Vector Start Upper / FPB MEM High Vector Start Lower, then dividing this according to the value of FPB MEM High Vector Granularity to determine the bit index within the vector.
 - If the bit index value is greater than the length indicated by FPB MEM High Vector Size Supported, then the Memory address is out of range (above) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
 - if the bit value within the vector at the calculated bit index location is 1b, then the Memory address is associated with the Secondary side of the bridge, else the Memory address is associated with the Primary side of the bridge.

IMPLEMENTATION NOTE: FPB ADDRESS DECODING §

FPB uses a bit vector mechanism to decode ranges of Routing IDs, and Memory Addresses above and below 4 GB. A bridge supporting FPB contains the following for each resource type/range where it supports the use of FPB:

- A Bit vector
- A Start Address
- A Granularity

These are used by the bridge to determine if a given address is part of the range decoded by FPB as associated with the secondary side of the bridge. An address that is determined not to be associated with the secondary side of the bridge using either or both of the non-FPB decode mechanisms and the FPB decode mechanisms is (by default) associated with the primary side of the bridge. Here, when we use the term “associated” we mean, for example, that the bridge will apply the following handling to TLPs:

- Associated with Primary, Received at Primary → Unsupported Request (UR)
- Associated with Primary, Received at Secondary → Forward upstream
- Associated with Secondary, Received at Primary → Forward downstream
- Associated with Secondary, Received at Secondary → Unsupported Request (UR)

In FPB, every bit in the vector represents a range of resources, where the size of that range is determined by the selected granularity. If a bit in the vector is Set, it indicates that TLPs addressed to an address within the corresponding range are to be associated with the secondary side of the bridge. The specific range of resources each bit represents is dependent on the index of that bit, and the values in the Start Address & Granularity. The Start Address indicates the lowest address described by the bit vector. The Granularity indicates the size of the region that is represented by each bit. Each successive bit in the vector applies to the subsequent range, increasing with each bit according to the Granularity.

For example, consider a bridge using FPB to describe a MEM Low range. FPB MEM Low Vector Start has been set to FC0h, indicating that the range described by the bit vector starts at address FC00 0000. FPB MEM Low Vector Granularity has been set to 0000b, indicating that each bit represents a 1 MB range.

From these values we can determine that bit 0 of the vector represents a 1MB range starting at FC000 0000 (FC00 0000-FC0F FFFF), bit 1 represents FC10 0000-FC1F FFFF, etc.

Bits in the vector that are set to 0 indicate that the range is not included in the range described by FPB. In the above example, If bit 0 is Clear, packets addressed to anywhere between FC00 0000 and FC0F FFFF should not be routed to the secondary bus of the bridge due to FPB.

IMPLEMENTATION NOTE: HARDWARE AND SOFTWARE CONSIDERATIONS FOR FPB §

FPB is intended to address a class of issues with PCI/PCIe architecture that relate to resource allocation inefficiency. These issues can be categorized as “static” or “dynamic” use case scenarios, where static use cases refer to scenarios where resources are allocated at system boot and then typically not changed again, and dynamic use cases refer to scenarios where run-time resource rebalancing (e.g., allocation of new resources, freeing of resources no longer needed) is required, due to hot add/remove, or by other needs.

In the Static cases there are limits on the size of hierarchies and number of Endpoints due to the use of additional Bus Numbers and the lack of use of Device Numbers caused by the PCI/PCIe architectural definition for Switches and Downstream Ports. FPB addresses this class of problems by “flattening” the use of Routing IDs (RIDs) so that Switches and Downstream Ports are able to make more efficient use of the available RIDs.

For the Dynamic cases, without FPB, the “best known method” to avoid rebalancing has been to reserve large ranges of Bus Numbers and Memory Space in the bridge above the relevant Port or Endpoint such that hopefully any future needs can be satisfied within the pre-allocated ranges. This leads to potentially unused allocations, which makes the Routing ID issues worse, and in a resource constrained platform this approach is difficult to implement, even for relatively simple cases, where, for example, one might have an add-in card implementing a single Endpoint replaced by another add-in card that has a Switch and two Endpoints, so that although an initial allocation of just one Bus would have been sufficient, the initial allocation breaks immediately with the new add-in card.

For Memory Space the pre-allocation approach is problematic when hot-plugged Endpoints may require the allocation of Memory Space below 4 GB, which by its nature is a limited resource, which is quickly used up by pre-allocation of even relatively small amounts, and for which pre-allocation is unattractive because of the multiple system elements placing demands on system address space allocation below 4 GB.

FPB includes mechanisms to enable discontinuous resource range allocation/reallocation for both Requester IDs and Memory Space. The intent is to allow system software the ability to maintain resource “pools” which can be allocated (and freed back to) at run-time, without disrupting other operations in progress as is required with rebalancing.

To support the run time use of FPB by system software, FPB hardware implementations should avoid introducing stalls or other types of disruptions to transactions in flight, including during the times that system software is modifying the state of the FPB hardware. It is not, however, expected that hardware will attempt to identify cases where system software erroneously modifies the FPB configuration in a way that does affect transactions in flight. Just as with the non-FPB mechanisms, it is the responsibility of system software to ensure that system operation is not corrupted due to a reconfiguration operation.

It is not explicitly required that system firmware/software perform the enabling and/or disabling of FPB mechanisms in a particular sequence, however care should be taken to implement resource allocation operations in a hierarchy such that the hardware and software elements of the system are not corrupted or caused to fail.

6.27 Vital Product Data (VPD) §

Vital Product Data (VPD) is the information that uniquely defines items such as the hardware, software, and microcode elements of a system. The VPD provides the system with information on various FRUs (Field Replaceable Unit) including Part Number, Serial Number, and other detailed information. VPD also provides a mechanism for storing information such as performance and failure data on the device being monitored. The objective, from a system point of view, is to collect this information by reading it from the hardware, software, and microcode components.

Support of VPD within add-in cards is optional depending on the manufacturer. Though support of VPD is optional, add-in card manufacturers are encouraged to provide VPD due to its inherent benefits for the add-in card, system manufacturers, and for Plug and Play.

The mechanism for accessing VPD is documented in § [Section 7.9.18](#).

VPD for PCI Express is unchanged from the definition in the [\[PCI-3.0\]](#). That definition, in turn, was based on earlier versions of the [\[PCI\]](#) as well as the [\[PLUG-PLAY-ISA-1.0a\]](#).

Vital Product Data is made up of Small and Large Resource Data Types.

Table 6-19 Small Resource Data Type Tag Bit Definitions §

| Offset | Field Name | | |
|--------------|--------------------|---------------------|------|
| Byte 0 | Value = 0xxx xybb | | |
| | Bit 7 | Small Resource Type | 0b |
| | Bits 6:3 | Small Item Name | xxxx |
| | Bits 2:0 | Length in bytes | yy |
| Bytes 1 to n | Actual information | | |

Table 6-20 Large Resource Data Type Tag Bit Definitions §

| Offset | Field Name | | |
|--------------|--|---------------------|----------|
| Byte 0 | Value = 1xxx xxxx b | | |
| | Bit 7 | Large Resource Type | 1b |
| | Bits 6:0 | Large Item Name | xxxxxxxx |
| Byte 1 | Length in bytes of data items bits[7:0] (lsb) | | |
| Byte 2 | Length in bytes of data items bits[15:8] (msb) | | |
| Bytes 3 to n | Actual data items | | |

The first VPD tag is the Identifier String (02h) and provides the product name of the device.

One VPD-R (10h) tag is used as a header for the read-only keywords. ~~↓↓The VPD-R list (including tag and length) must checksum to zero.↓~~ Attempts to write the read-only data will be executed as a no-op.

Errata: Base 6.3
B803△◀

One VPD-W (11h) tag is used as a header for the read-write keywords. The storage component containing the read/write data is a non-volatile device that will retain the data when powered off.

The last tag must be the End Tag (0Fh).

A small example of the resource data type tags used in a typical VPD is shown in § [Table 6-21](#).

Table 6-21 Resource Data Type Flags for a Typical VPD §

| | | |
|---------|--|--------------------------|
| TAG 02h | Identifier String | Large Resource Data Type |
| TAG 10h | VPD-R list containing one or more VPD keywords | Large Resource Data Type |
| TAG 11h | VPD-W list containing one or more VPD keywords | Large Resource Data Type |

| | | |
|---------|---------|--------------------------|
| TAG 0Fh | End Tag | Small Resource Data Type |
|---------|---------|--------------------------|

6.27.1 VPD Format §

Information fields within a VPD resource type consist of a three-byte header followed by some amount of data (see § Figure 6-34). The three-byte header contains a two-byte keyword and a one-byte length. A keyword is a two-character (ASCII) mnemonic that uniquely identifies the information in the field. The last byte of the header is binary and represents the length value (in bytes) of the data that follows.

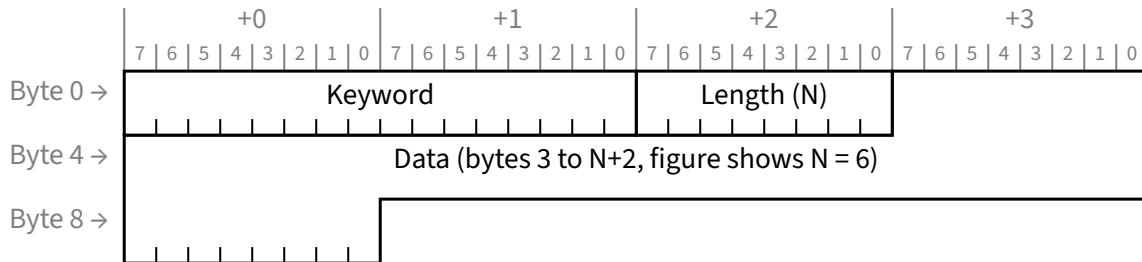


Figure 6-34 VPD Format §

VPD keywords are listed in two categories: read-only fields and read/write fields. Unless otherwise noted, keyword data fields are provided as ASCII characters. Use of ASCII allows keyword data to be moved across different enterprise computer systems without translation difficulty.

An example of the “add-in card serial number” VPD item is as follows:

Table 6-22 Example of Add-in Serial Card Number §

| | | |
|---------|---------|------------------|
| Byte 0 | 53h “S” | Keyword: SN |
| Byte 1 | 4Eh “N” | |
| Byte 3 | 08h | Length: 8 |
| Byte 4 | 30h “0” | Data: “00000194” |
| Byte 5 | 30h “0” | |
| Byte 6 | 30h “0” | |
| Byte 7 | 30h “0” | |
| Byte 8 | 30h “0” | |
| Byte 9 | 31h “1” | |
| Byte 10 | 39h “9” | |
| Byte 11 | 34h “4” | |

6.27.2 VPD Definitions §

This section describes the current VPD large and small resource data tags plus the VPD keywords. This list may be enhanced at any time. Companies wishing to define a new keyword should contact the PCISIG. All unspecified values are reserved for SIG assignment.

6.27.2.1 VPD Large and Small Resource Data Tags §

VPD is contained in four types of Large and Small Resource Data Tags. The following tags and VPD keyword fields may be provided in PCI devices.

Table 6-23 VPD Large and Small Resource Data Tags §

| Tag | Description |
|---|--|
| Large resource type Identifier String Tag (02h) | This tag is the first item in the VPD storage component. It contains the name of the add-in card in alphanumeric characters. |
| Large resource type VPD-R Tag (10h) | This tag contains the read only VPD keywords for an add-in card. |
| Large resource type VPD-W Tag (11h) | This tag contains the read/write VPD keywords for an add-in card. |
| Small resource type End Tag (0Fh) | This tag identifies the end of VPD in the storage component. |

6.27.2.2 Read-Only Fields §

Table 6-24 VPD Read-Only Fields §

| Keyword | Name | Description |
|---------|---|---|
| PN | Add-in Card Part Number | This keyword is provided as an extension to the Device ID (or Subsystem ID) in the Configuration Space header in § Figure 6-34 . |
| EC | Engineering Change Level of the Add-in Card | The characters are alphanumeric and represent the engineering change level for this add-in card. |
| FG | Fabric Geography | Reserved for legacy use by [PICMG] specifications. |
| LC | Location | Reserved for legacy use by [PICMG] specifications. |
| MN | Manufacture ID | This keyword is provided as an extension to the Vendor ID (or Subsystem Vendor ID) in the Configuration Space header in § Figure 6-34 . This allows vendors the flexibility to identify an additional level of detail pertaining to the sourcing of this device. |
| PG | PCI Geography | Reserved for legacy use by [PICMG] specifications. |
| SN | Serial Number | The characters are alphanumeric and represent the unique add-in card Serial Number. |

Base 6.4 vs Base 6.3

| Keyword | Name | Description |
|---------|-----------------------|--|
| TR | Thermal Reporting | <p>This keyword provides a standard interface for reporting four fields: AFI Level, MaxTherm, DTherm, and MaxAmbient. The data area for this field is four bytes long. This data is encoded as a 4-byte binary value in little endian order (byte 0 contains bits 7:0). This value contains the four fields as follows: AFI Level bits [3:0], MaxTherm bits [7:4], DTherm bits [11:8], and MaxAmbient bits [19:12] are placed in bits 19:0. Bits 31:20 are Reserved and must be set to 000h. Field description is provided within the [CEM]. This keyword is intended to be used only in designs based on that form factor specification.</p> <p>Note that due to the character nature of the VPD encoding mechanism, this binary value is permitted to start on any byte boundary within the VPD.</p> |
| Vx | Vendor Specific | <p>This is a vendor specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 or A through Z.</p> |
| CP | Extended Capability | <p>This field allows a new capability to be identified in the VPD area. Since dynamic control/status cannot be placed in VPD, the data for this field identifies where, in the device's memory or I/O address space, the control/status registers for the capability can be found. Location of the control/status registers is identified by providing the index (a value between 0 and 5) of the Base Address register that defines the address range that contains the registers, and the offset within that Base Address register range where the control/status registers reside. The data area for this field is four bytes long. The first byte contains the ID of the extended capability. The second byte contains the index (zero based) of the Base Address register used. The next two bytes contain the offset (in little-endian order) within that address range where the control/status registers defined for that capability reside.</p> |
| RV | Checksum and Reserved | <p>The first byte of this item is a checksum byte. The checksum is correct if the sum of all bytes in VPD (from VPD address 0 up to and including this byte) is zero. The remainder of this item is reserved space (as needed) to identify the last byte of read-only space. The read-write area does not have a checksum. This field is required.</p> |
| FF | Form Factor | <p>This keyword indicates a string that identifies the form factor and version associated with this add-in-card. Values are a lower case string. The string consists of a list of one or more elements separated by colons (“ : ”).</p> <p>The first element is a sequence of lowercase strings separated by periods that identifies the specification(s) associated with the form factor. The first element string is the reserved domain name associated with the authority defining that form factor (i.e., like those used in [DNS] records), followed by one or more strings that identify a particular form factor, followed by a Version Number for that form factor specification. Any characters in the form factor name other than “ a ” to “ z ”, “ 0 ” to “ 9 ”, “ * ”, “ ? ”, and “ – ” are dropped (e.g., M.2 becomes m2). The character “ * ” represents a wild card (arbitrary characters, including “ . ”). The value “ – ” is used to indicate no form factor. The value “ ? ” or the absence of the FF keyword is used to indicate that the form factor is unknown.</p> <p>Subsequent elements are optional and contain attributes describing variations of that form factor (e.g., size, connector, keying, ...). These elements are defined by the indicated form-factor Specification and consist of either an “ option ” string or a “ key=value ” string. Valid characters in the “ option ” or “ key ” portion are “ a ” to “ z ”, “ 0 ” to “ 9 ”, “ * ”, “ ? ”, and “ – ”. The “ value ” portion may contain any character other than the colon “ : ”. The order of attributes is not significant. Attributes are not permitted when the first element is “ – ” or “ ? ” (no form factor or unknown form factor).</p> <p>Examples are:</p> <ul style="list-style-type: none"> • com.pcisig.cem.4.0 • com.pcisig.cem.* • com.pcisig.m2.1.0:2280 • com.pcisig.m2.1.0:size=2280:key=M • com.pcisig.m2.1.0:bga |

| Keyword | Name | Description |
|---------|------|--|
| | | <ul style="list-style-type: none"> • org.snia.sff-ta-1001.1.0.2 <p>A given add-in card is permitted to claim conformance to multiple form factor specifications. This is indicated by using the wild card character “*” or by using multiple FF fields (which in turn could use the wild card character).</p> <p>When the form factor of a slot does not match some FF keyword of the add-in card in that slot, this indicates the presence of one or more “Carrier Cards” to convert power and sideband signals of the slot to those of the add-in card. The mechanism(s) used to identify a particular Carrier Card and to describe how it operates are outside the scope of this specification.</p> |

6.27.2.3 Read/Write Fields §

Table 6-25 VPD Read/Write Fields §

| Keyword | Name | Description |
|---------|----------------------------|--|
| Vx | Vendor Specific | This is a vendor specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 or A through Z. |
| Yx | System Specific | This is a system specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 or B through Z. |
| YA | Asset Tag Identifier | This is a system specific item and the characters are alphanumeric. This keyword contains the system asset identifier provided by the system owner. |
| RW | Remaining Read/ Write Area | This descriptor is used to identify the unused portion of the read/write space. The product vendor initializes this parameter based on the size of the read/write space or the space remaining following the Vx VPD items. One or more of the Vx, Yx, and RW items are required. |

6.27.2.4 VPD Example §

The following is an example of a typical VPD.

Table 6-26 VPD Example §

| Offset | Item Value |
|--------|--|
| 0 | Large Resource Type ID String Tag (02h) 82h “Product Name” |
| 1 | Length 0021h |
| 3 | Data “ABCD Super-Fast Widget Controller” |
| 36 | Large Resource Type VPD-R Tag (10h) 90h |
| 37 | Length 0059h |
| 39 | VPD Keyword “PN” |
| 41 | Length 08h |
| 42 | Data “6181682A” |
| 50 | VPD Keyword “EC” |

Base 6.4 vs Base 6.3

| Offset | Item Value |
|--------|---|
| 52 | Length 0Ah |
| 53 | Data “4950262536” |
| 63 | VPD Keyword “SN” |
| 65 | Length 08h |
| 66 | Data “00000194” |
| 74 | VPD Keyword “MN” |
| 76 | Length 04h |
| 77 | Data “1037” |
| 81 | VPD Keyword “RV” |
| 83 | Length 2Ch |
| 84 | Data Checksum |
| 85 | Data Reserved (00h) |
| 128 | Large Resource Type VPD-W Tag (11h) 91h |
| 129 | Length 007Ch |
| 131 | VPD Keyword “V1” |
| 133 | Length 05h |
| 134 | Data “65A01” |
| 139 | VPD Keyword “Y1” |
| 141 | Length 0Dh |
| 142 | Data “Error Code 26” |
| 155 | VPD Keyword “RW” |
| 157 | Length 61h |
| 158 | Data Reserved (00h) |
| 255 | Small Resource Type End Tag (0Fh) 78h |

6.28 Native PCIe Enclosure Management §

NPEM is an optional PCIe Extended Capability that provides mechanisms for enclosure management. This mechanism is designed to provide management for enclosures containing PCIe SSDs that is consistent with the established capabilities in the storage ecosystem.

This section defines the architectural aspects of the mechanism. The NPEM extended capability is defined in § [Section 7.9.19](#).

An enclosure is any platform, box, rack, or set of boxes that contain one or more PCIe SSDs. The NPEM capability provides storage related enclosure control (e.g., status LED control) for a PCIe SSD. The NPEM capability may reside in a Downstream Port, or an Endpoint (i.e., the PCIe SSD). § Figure 6-35 shows an example configuration with a single Downstream Port containing the NPEM capability and vendor specific logic to control the associated LEDs.

Base 6.4 vs Base 6.3

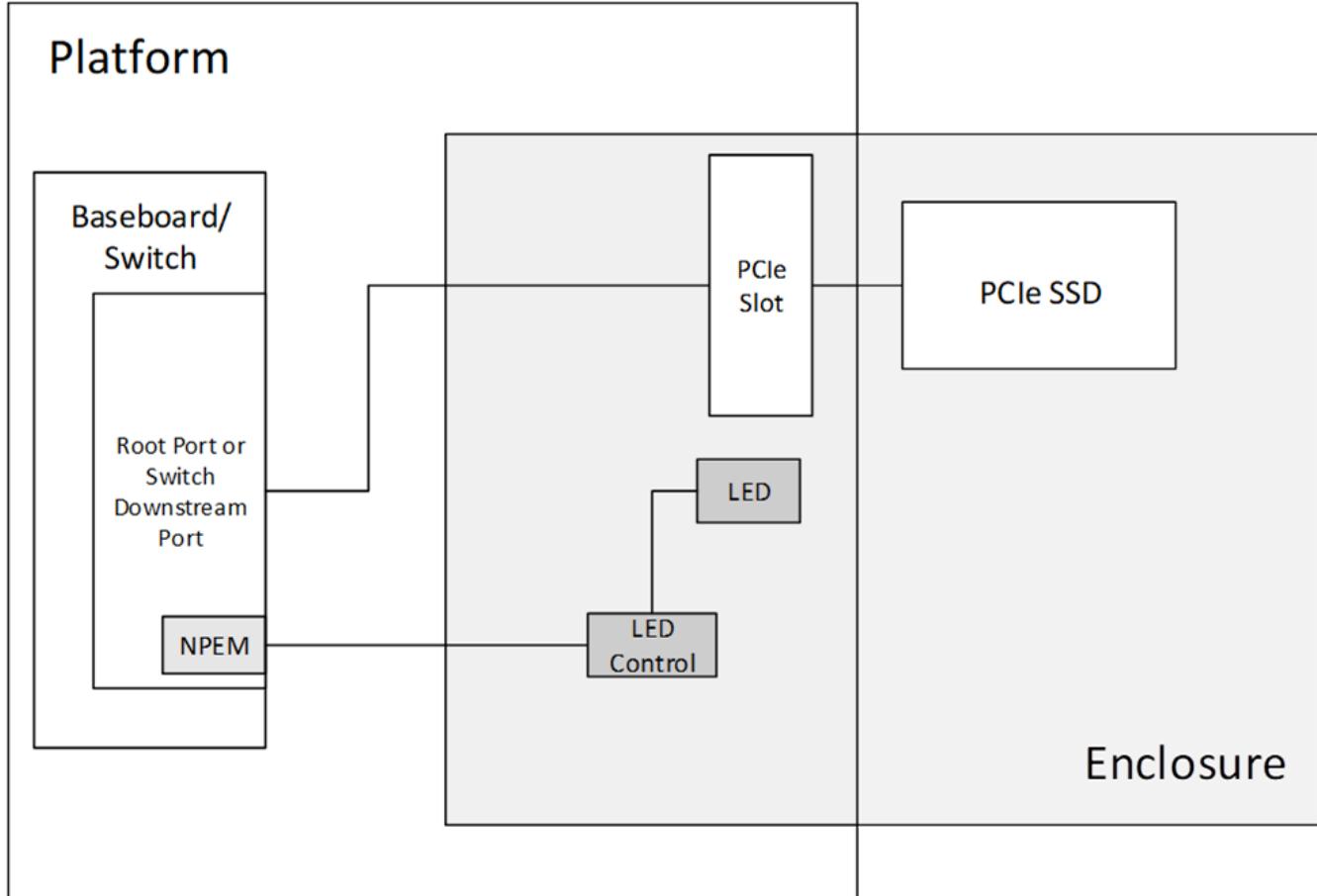


Figure 6-35 Example NPEM Configuration using a Downstream Port §

§ Figure 6-36 shows an example configuration with the NPEM capability located in the Upstream Port (in this case, the SSD function).

Base 6.4 vs Base 6.3

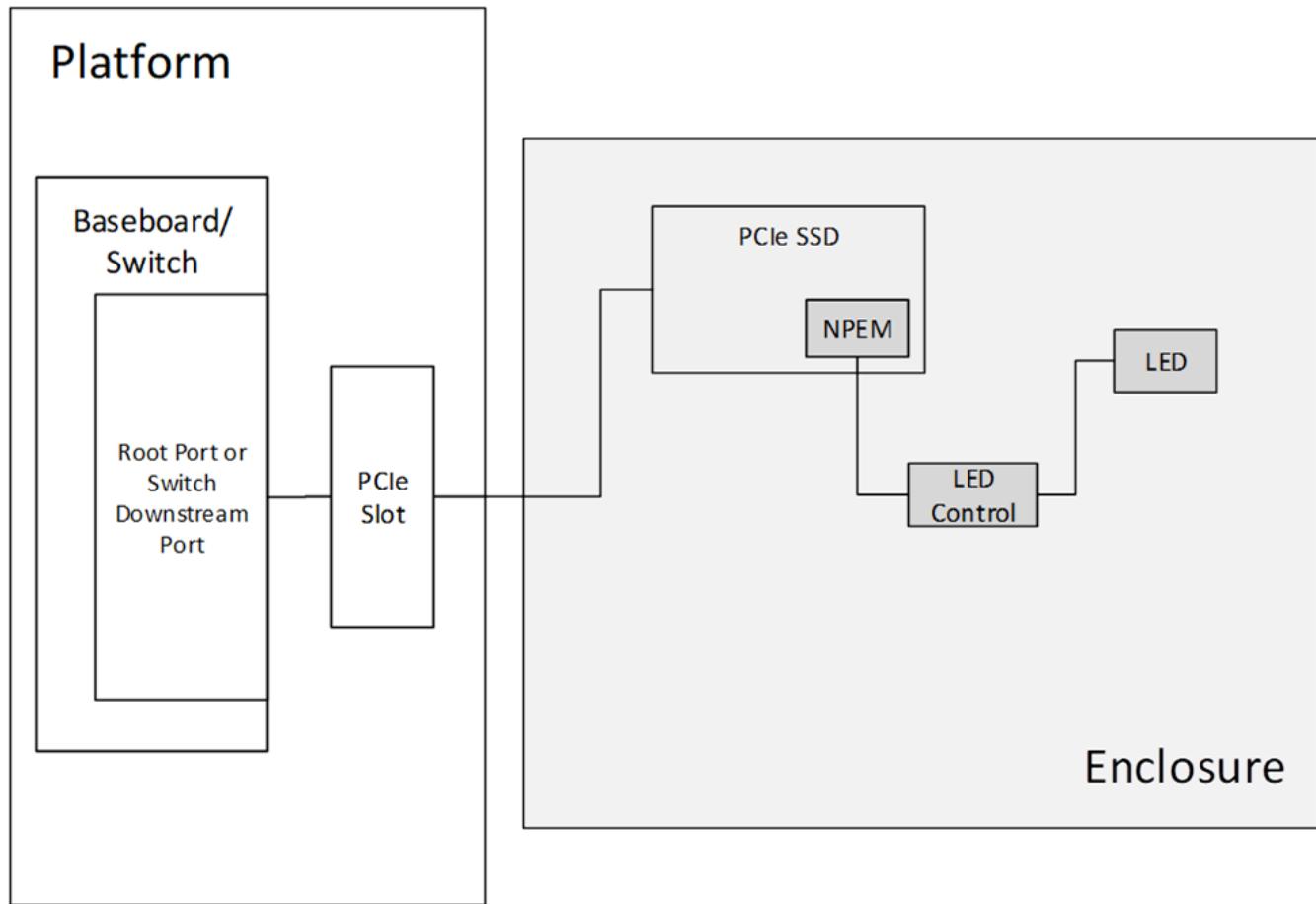


Figure 6-36 Example NPEM Configuration using an Upstream Port §

Software issues an NPEM command by writing to the NPEM Control register to change the indications associated with an SSD. NPEM Command is a single write to the NPEM Control register that changes the state of zero or more bits. NPEM indicates a successful completion to software using the command completed mechanism. § Figure 6-37 shows the overall flow.

This specification defines the software interface provided by the NPEM capability. The Port to enclosure interface, enclosure, enclosure to LED interface, number of LEDs per SSD, and associated LED blink patterns are all outside the scope of this specification.

Base 6.4 vs Base 6.3

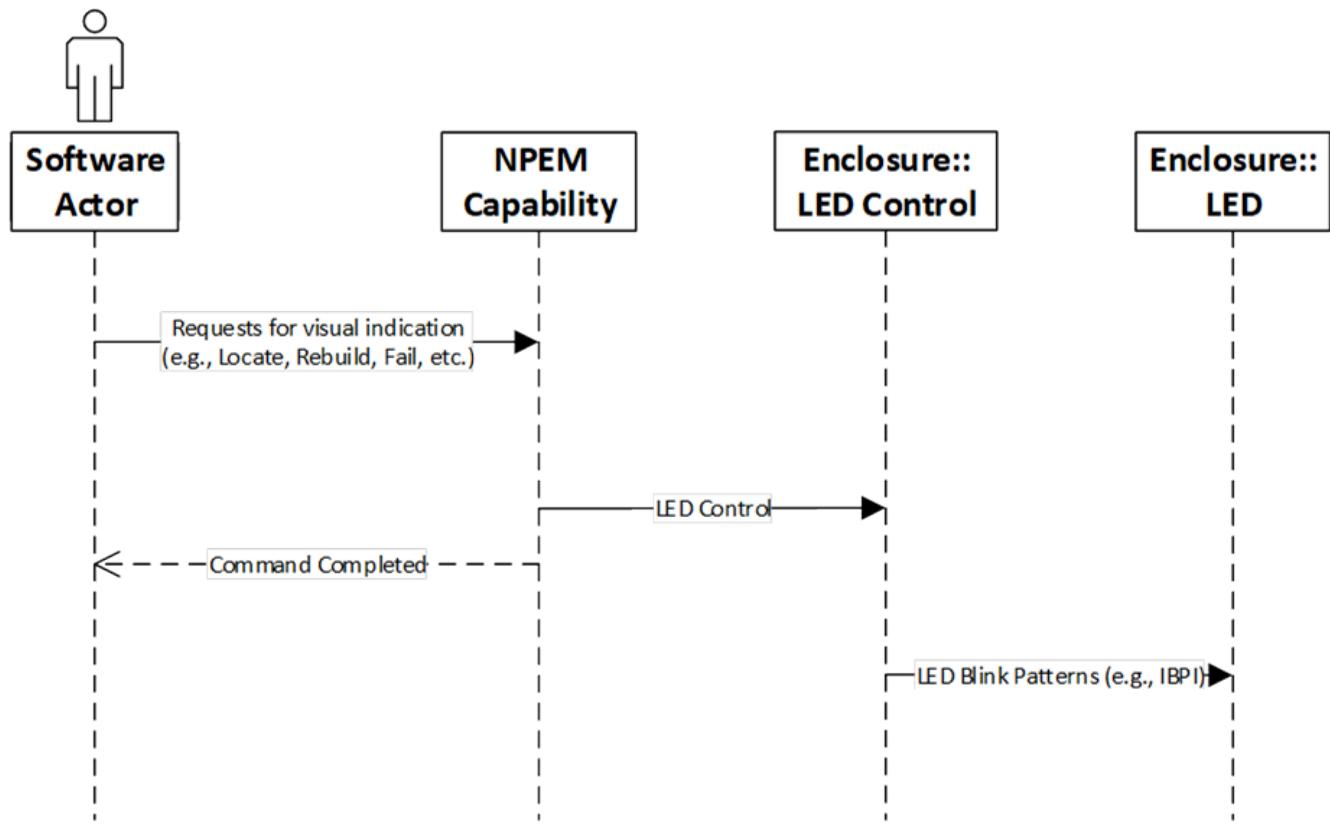


Figure 6-37 NPEM Command Flow §

NPEM provides a mechanism for system software to issue a reset to the LED control element within the enclosure by means of the NPEM Reset mechanism, which is independent of the PCIe Link itself. The NPEM command completed mechanism also applies to NPEM Reset.

Storage system admin or software controls the indications for various device states through the NPEM capability.

IMPLEMENTATION NOTE: NPEM STATES §

§ Table 6-27 shows an example of NPEM states and a possible meaning that some enclosures may assign to the architected NPEM states.

Table 6-27 NPEM States §

| NPEM State | Actor | Definition |
|---------------------|----------------------------------|--|
| OK | System Admin or Storage Software | OK state may mean the drive is functioning normally. This state may implicitly mean that an SSD is present, powered on, and working normally as seen by the software. A more granular indication of drive not physically present or present but not powered up are both outside the scope of this specification. |
| Locate | System Admin | Locate state may mean the specific drive is being identified by an admin. |
| Fail | Storage Software | Fail state may mean the drive is not functioning properly |
| Rebuild | Storage Software | Rebuild state may mean this drive is part of a multi-drive storage volume/array that is rebuilding or reconstructing data from redundancy on to this specific drive. |
| PFA | Storage Software | PFA stands for Predicted Failure Analysis. This state may mean the drive is still functioning normally but predicted to fail soon. |
| Hot Spare | Storage Software | Hot Spare state may mean this drive is marked to be automatically used as a replacement for a failed drive and contents of the failed drive may be rebuilt on this drive. |
| In A Critical Array | Storage Software | In A Critical Array state may mean the drive is part of a multi-drive storage array and that array is degraded. |
| In A Failed Array | Storage Software | NPEM In A Failed Array state may mean the drive is part of a multi-drive storage array and that array is failed. |
| Invalid Device Type | Storage Software | Invalid Device Type state may mean the drive is not the right type for the connector (e.g., An enclosure supports SAS and NVMe drives and this drive state indicates that a SAS drive is plugged into an NVMe slot). |
| Disabled | Storage Software | Disabled state may mean the drive in this slot is disabled. A removal of this drive from the slot may be safe. The power from this slot may be removed. |

IMPLEMENTATION NOTE: SOFTWARE POLLING OF NPEM COMMAND COMPLETED §

Different NPEM implementations may vary widely in how long they take to complete NPEM commands, from instantaneous to tens of ms. To avoid or minimize software polling overheads, it is recommended that software implement one or both of the following optimizations.

Instead of software writing a command and then immediately polling for completion, it is recommended that software reverse this order. When ready to write a new command, software first polls for completion of the previous command, and then writes the new command. This enables overlapped operation, often completely hiding the time it takes hardware to execute an NPEM command. To enable this polling model, software must initialize the hardware following a reset by writing a no-op command in order to have hardware generate the first NPEM command completion.

For the case where an NPEM command has not completed when software polls the bit, it is recommended that software not continuously “spin” on polling the bit, but rather poll under interrupt at a reduced rate; for example at 10 ms intervals.

6.29 Conventional PCI Advanced Features Operation §

For Conventional PCI devices integrated into a Root Complex, the Conventional PCI Advanced Features Capability (AF) provides mechanisms for using advanced features originally developed for PCI Express.

- The Function Level Reset (INITIATE_FLR) mechanism enables software to quiesce and reset hardware with Function-level granularity.

FLR applies on a per Function basis. Only the targeted Function is affected by the FLR operation.

- The ~~↓↑Transactions Pending↓↑Transactions Pending (TP)↑↓↑(TP)↓~~ mechanism is used to indicate that the Function ~~↓↑has issued↓↑is expecting↓↑one or more↓↑non-posted transactions (including Delayed Transactions) which have not been completed.↓↑Completions.↑~~

Errata: Base 6.3
B817△◀▷

The FLR and TP mechanisms defined here are strictly for Conventional PCI devices integrated into a Root Complex where the implementation permits non-posted transactions for a given Conventional PCI Function to complete even if the value of the Bus Master Enable bit in its Command Register is 0b. Implementations that do not meet this requirement must not implement the FLR and TP mechanisms.

FLR modifies the Function state as follows:

Function registers and Function-specific state machines must be set to their initialization values as specified in this document, except for the following bits, which must not be modified: Fast Back-to-Back Transactions Enable, Cache Line Size, Latency Timer, Interrupt Line, PME_En, PME_Status.

Note that the controls that enable the Function to initiate bus transactions are cleared, including the Bus Master Enable bit in the Command Register, the MSI Enable bit in the MSI Capability Structure, and the like, effectively causing the Function to become quiescent.

After an FLR has been initiated, the Function must complete the FLR within 100 ms. If software initiates an FLR when the Transactions Pending bit is 1b, then software must not initialize the Function until allowing adequate time to achieve reasonable certainty that any outstanding transactions will have completed. The Transactions Pending bit must be clear upon completion of the FLR.

FLR modifies Function state not described by this specification (in addition to state that is described by this specification), and so the following criteria must be applied using Function- specific knowledge to evaluate the Function's behavior in response to an FLR :

- The Function must not give the appearance of an initialized adapter with an active host on any external interfaces controlled by that Function. The steps needed to terminate activity on external interfaces are outside of the scope of this specification.
 - For example, a network adapter must not respond to queries that would require adapter initialization by the host system or interaction with an active host system, but is permitted to perform actions that it is designed to perform without requiring host initialization or interaction. If the network adapter includes multiple Functions that operate on the same external network interface, this rule affects only those aspects associated with the particular Function reset by FLR .
- The Function must not retain within itself software readable state that potentially includes secret information associated with any preceding use of the Function. Main host memory assigned to the Function must not be modified by the Function.
 - For example, a Function with internal memory readable directly or indirectly by host software must clear or randomize that memory.
- The Function must return to a state such that normal configuration of the Function's PCI interface will cause it to be $\downarrow\downarrow$ useable $\downarrow\downarrow$ $\uparrow\uparrow$ usable \uparrow by drivers normally associated with the $\downarrow\downarrow$ Function $\downarrow\downarrow$ $\uparrow\uparrow$ Function. \uparrow

When an FLR is initiated, the targeted Function must behave as follows:

- The Function must complete normally the configuration write that initiated the FLR operation and then initiate the FLR .
- While an FLR is in progress:
 - The Function must not respond to any request on the bus (i.e., requests targeting the Function will Master Abort).

The $\downarrow\downarrow$ Transactions Pending $\downarrow\downarrow$ $\uparrow\uparrow$ Transactions Pending (TP) $\uparrow\uparrow$ $\downarrow\downarrow$ (TP) $\downarrow\downarrow$ bit indicates that the Function $\downarrow\downarrow$ has issued $\downarrow\downarrow$ $\uparrow\uparrow$ is expecting $\uparrow\uparrow$ one or more $\downarrow\downarrow$ non posted transactions which have not been completed. $\downarrow\downarrow$ $\uparrow\uparrow$ Completions. $\uparrow\uparrow$ This field may be used by software to determine when a Function has become quiescent.

Errata: Base 6.3
B817 Δ \triangleleft \triangleright

IMPLEMENTATION NOTE: AVOIDING ISSUES WITH PENDING TRANSACTIONS §

An FLR causes a Function to lose track of any pending (outstanding ~~↑↓non-posted~~)
~~↑↑Non-Posted and UIO~~ transactions. Depending upon the specific implementation of
the RC-integrated PCI Function, if software issues an FLR while there are pending
transactions, there is a possibility for data corruption as described in the “[Avoiding Data Corruption From Stale
Completions](#)” Implementation Note.

Errata: Base 6.3
B834△↔

To avoid potential issues with Root Complex implementations where Stale Completions are possible or a Discard Timer is present, it is recommended that software use an algorithm similar to the following:

1. Software that's performing the FLR synchronizes with other software that might potentially access the Function directly, and ensures that such accesses will not occur during this algorithm.
2. Software clears the entire Command register, disabling the Function from mastering any new transactions.
3. Software polls the Transactions Pending bit in the AF Status Register either until it's clear or until it's been long enough to achieve reasonable certainty that any remaining outstanding Transactions will never complete. On many systems, the Transactions Pending bit will usually clear within a few milliseconds, so software might choose to poll during this initial period using a tight software loop. On rare cases when the Transactions Pending bit doesn't clear by this time, software will need to poll for a longer system-specific period (potentially seconds), so software might choose to conduct this polling using a timer-based interrupt polling mechanism.
4. Software initiates the FLR.
5. Software waits 100 ms.
6. Software reconfigures the Function and enables it for normal operation.

6.30 Data Object Exchange (DOE) §

Data Object Exchange (DOE) is an optional mechanism for system firmware/software to perform data object exchanges with a Function or RCRB. Software discovers DOE support via the Data Object Exchange (DOE) Extended Capability structure. Because DOE depends on Configuration Requests it is not usable for peer-to-peer operations directly between Functions, although system software can provide a mechanism to relay data objects between Functions if such capabilities are desired. When DOE is implemented in an RCRB, it is permitted to block peer-to-peer operation via implementation specific means.

DOE is a prerequisite Extended Capability for a Function to support in-band access by system firmware/software using Configuration Requests to Component Measurement and Authentication (CMA). CMA in turn builds on [SPDM].

It is permitted to implement DOE in any type of Function, and in an RCRB. It is permitted to implement DOE more than once in a single Function or RCRB.

DOE uses the terms “type” and “feature” with specific meanings. A data object type indicates a group of one or more data objects, and can be used to determine how to further construct or parse a specific data object. Every data object includes an indication of its type. Data object types are defined by particular vendors and distinguished by Vendor ID. It is permitted for features/types defined by one vendor to require the use of features/types defined by another vendor.

A feature is a capability of some sort that uses data objects. A feature can be associated with one or more data object types. Each specific feature defines the mechanism(s) used by software to discover support for the feature, for example by means of a capabilities bit in Configuration Space, or implied by the presence of an associated Extended Capability structure. A feature using data objects must specify the scope of the specific features in relation to the Function(s) implementing that feature via DOE, for example if only the Function itself is associated with the feature, or if Function 0 of a Multi-Function Device represents the Device as a whole, etc.

It is permitted, but not recommended, for a feature using data object exchanges to require that a Function implement a unique instance of DOE for that specific feature, and/or to allow sharing of a DOE instance to only a specific set of features using data object exchange, and/or to allow a Function to implement multiple instances of DOE supporting the specific feature.

It is permitted to use DOE when a Function is in non-D0 states, although it is permitted for a specific data object feature to restrict operation in non-D0 states.

IMPLEMENTATION NOTE: SUPPORTING MULTIPLE HARDWARE/SOFTWARE BINDINGS §

When multiple features are supported by a device, or when a given feature can be used by more than one software entity at a time, it is typically necessary to coordinate the use of the hardware/software interface between entities implementing the multiple features. SPDM Version 1.2.0 introduced the SPDM connection model, addressing some important needs, but not itself addressing the hardware/software interface problems arising when multiple software entities need to maintain different contexts and/or configurations. DOE provides two basic mechanisms for addressing this type of issue.

One option is that specific data object features may require the use of dedicated instances of DOE, particularly to allow system software to assign ownership of a specific DOE instance with the software entity using the specific data object feature(s) associated with that particular instance. For example, data object features A and B may perform different tasks, and so by instantiating dedicated instances of DOE for each, system software avoids the need to implement ownership control and arbitration mechanisms between A and B.

When this is done, if the underlying hardware uses shared resources to implement A and B, then the hardware may require the ability to maintain separate contexts for each data object feature, because system software will allow the two features to be operated at the same time.

Alternatively, for CMA/SPDM and related use cases, the optional Connection ID mechanism can eliminate the need for multiple DOE instances. Typically, the Connection ID mechanism provides much better scaling, and it is recommended that the Connection ID mechanism be used where applicable, instead of requiring the use of dedicated DOE instances. § Figure 6-38 illustrates a “stack diagram” where six applications are all using CMA/SPDM over DOE in three groups.

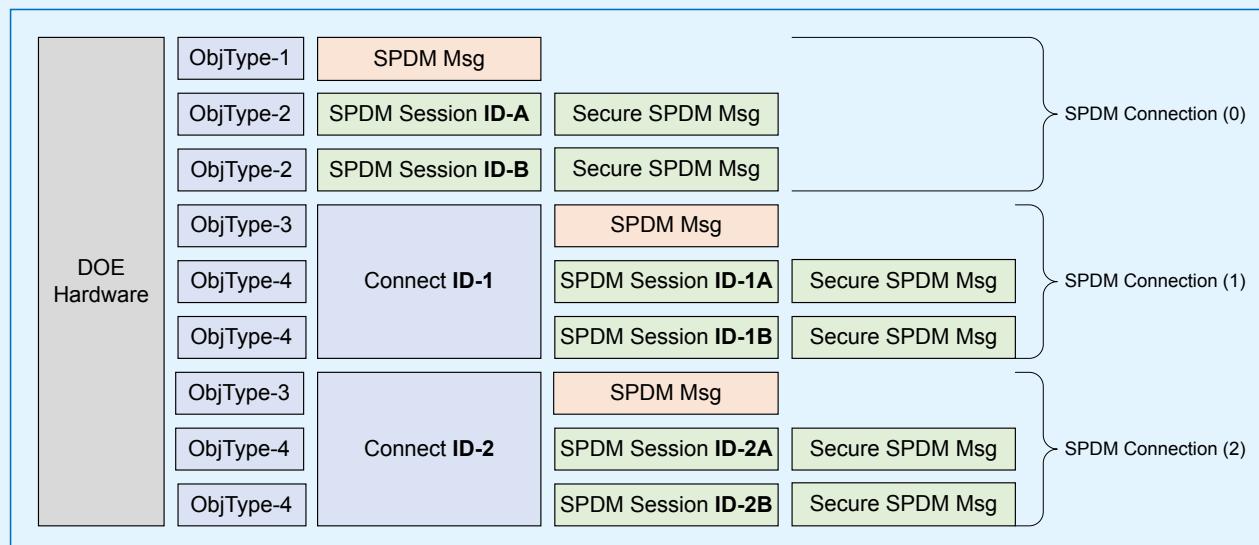


Figure 6-38 Stack Diagram Illustration of Multiple Sessions and Connections §

In each case, two distinct SPDM uses are shown, A and B, and within each group, the SPDM configuration must be the same, but each group may have distinct requirements from the others, e.g., different hashing algorithms. The SPDM Driver and PCI Driver software components must coordinate the establishment of each connection, including the assignment of a Connection ID, and, at run time, must manage the transfer of data objects across

the DOE interface itself to ensure that multiple software entities cannot attempt to access the DOE Mailbox registers during the transfer of a single data object. However, context is maintained in the device for each session. § Figure 6-39 illustrates at a high level how these elements relate to each other.

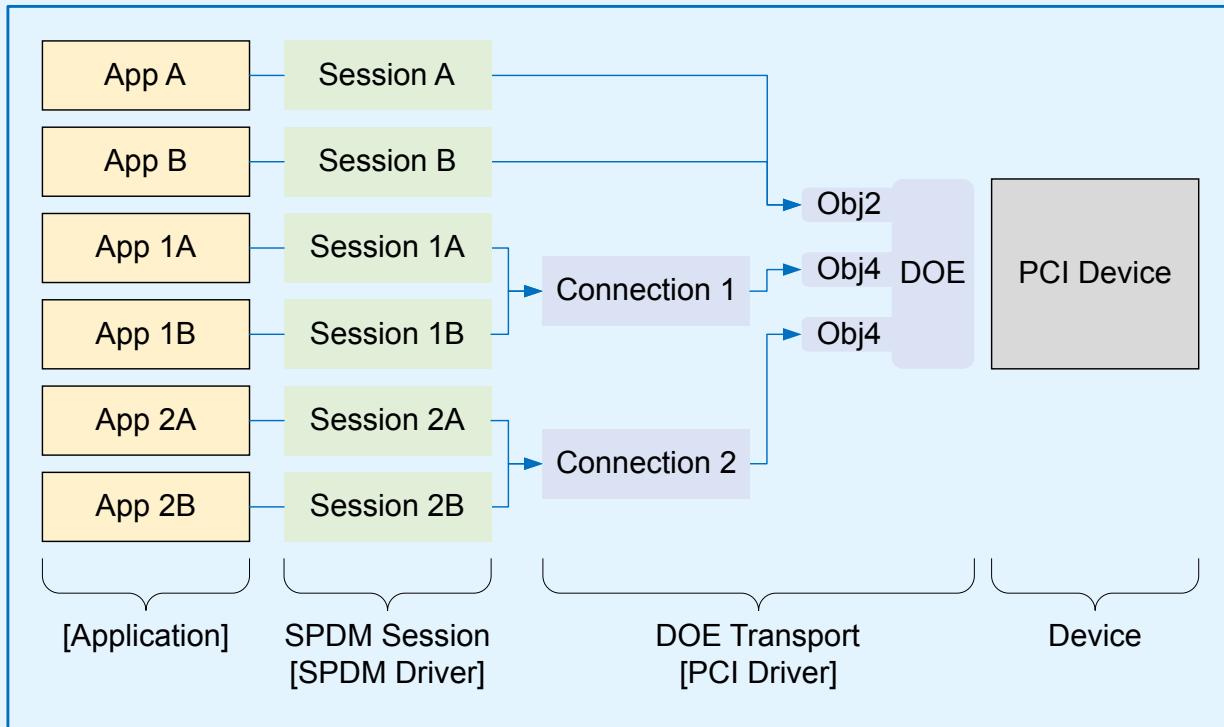


Figure 6-39 Example Showing Relationships of Software and Hardware Elements §

6.30.1 Data Objects and Features §

Data objects must consist of 2 DW to 256K DW, as shown in § Figure 6-40 .

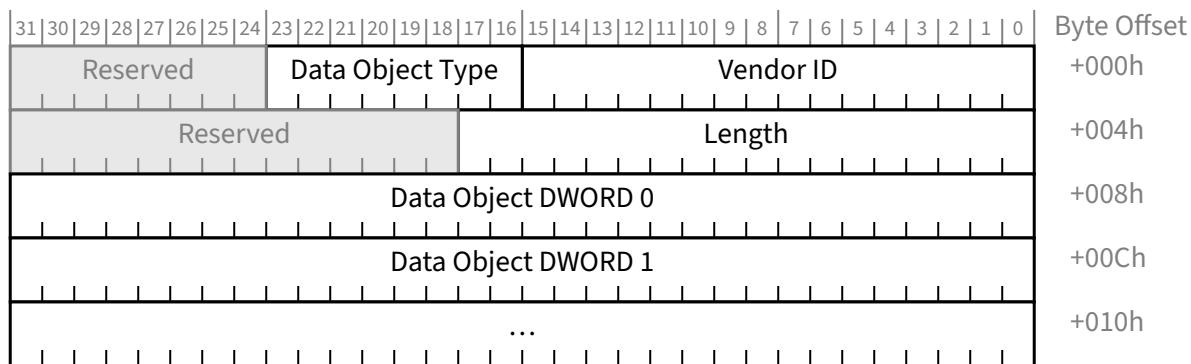


Figure 6-40 DOE Data Object Format §

The first DW of a data object must be formatted as defined in § Table 6-28 and illustrated in § Figure 6-41 .

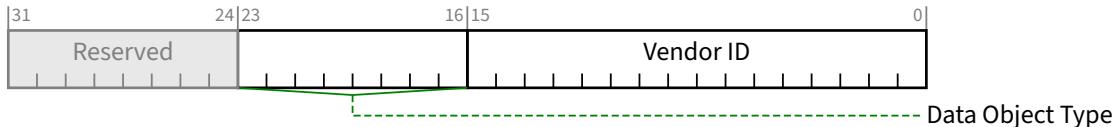


Figure 6-41 DOE Data Object Header 1 §

Table 6-28 DOE Data Object Header 1 §

| Bit Location | Description |
|--------------|--|
| 15:0 | Vendor ID - PCI-SIG Vendor ID of the entity that defined the type of data object. |
| 23:16 | Data Object Type – The type of data object. |

The Second DW of a data object must be formatted as defined in § Table 6-29 and illustrated in § Figure 6-42 .

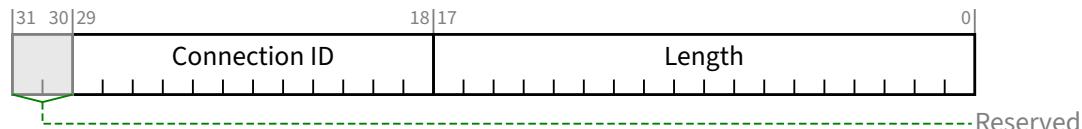


Figure 6-42 DOE Data Object Header 2 §

Table 6-29 DOE Data Object Header 2 §

| Bit Location | Description |
|--------------|--|
| 17:0 | Length – Length of the data object being transferred in number of DWs including the 2 DWs of the Data Object Header, encoded such that a value of 0 0002h indicates 2 DWs, and a value of 0 0000h indicates 2 ¹⁸ DWs. A value of 0 0001h is not allowed. |
| 29:18 | Connection ID – For data object types using Connection ID, this field contains the Connection ID. For all other data object types, this field is Reserved. |

Each data object is uniquely identified by the Vendor ID of the vendor publishing the data object definition and a Data Object Type value assigned by that vendor (see § Table 6-28). See § Table 6-33 for a list of data objects defined by PCI-SIG. Data objects of a specific type are permitted to vary in size and composition.

Unless a data object type definition specifies a different requirement, the following rules apply:

- If the number of DW transferred does not match the Length indicated in DOE Data Object Header 2 for a data object, then the entire data object must be silently discarded.
- If the Length indicated in DOE Data Object Header 2 is shorter than expected for a specific data object, then the data object must be silently discarded.

- If the Length indicated in DOE Data Object Header 2 is greater than expected for a specific data object, then the portion of the data object up to the expected length must be processed normally and the remainder of the data object must be silently discarded.
- If the data object type is not supported, then that data object must be silently discarded.

6.30.1.1 DOE Discovery Feature §

The DOE Discovery feature must be implemented, and provides a means for software to discover the data object types supported by a DOE instance. The DOE Discovery type consists of the request and response data objects, with the 3rd DW of the data object content as defined in § Table 6-30 and § Table 6-31 respectively. Where indicated in § Table 6-33, response data objects include a 4th DW, as defined in § Table 6-32. The DOE Discovery data object feature must be operable in D0, D1, D2 and D3_{hot}.

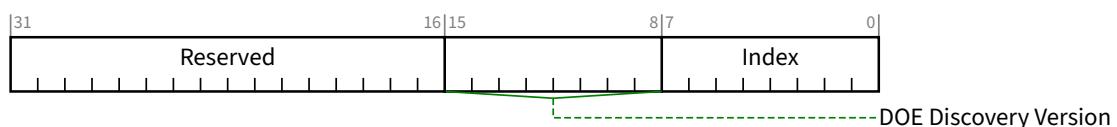


Figure 6-43 DOE Discovery Request Data Object Contents (3rd DW) §

Table 6-30 DOE Discovery Request Data Object Contents (3rd DW) §

| Bit Location | Description |
|--------------|---|
| 7:0 | Index – Indicates DOE Discovery entry index queried. Indices must start at 00h and increase monotonically by 1. |
| 15:8 | DOE Discovery Version – must be 02h if the Capability Version in the Data Object Exchange Extended Capability is 02h or greater. |
| 31:16 | Reserved – Requesters must place 0000h in this field. Responders must ignore the value in this field. |

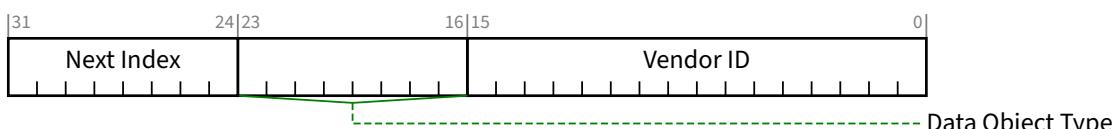


Figure 6-44 DOE Discovery Response Data Object Contents (3rd DW) §

Table 6-31 DOE Discovery Response Data Object Contents (3rd DW) §

| Bit Location | Description |
|--------------|---|
| 15:0 | Vendor ID – PCI-SIG Vendor ID of the entity that defined the type of data object. FFFFh if index is invalid or out of range. |
| 23:16 | Data Object Type – Indicates the identity of the data object type associated with the Index value supplied with the DOE Discovery Request. |

Base 6.4 vs Base 6.3

| Bit Location | Description |
|--------------|--|
| | <p>The PCI-SIG defined data object type for DOE Discovery must be implemented at index 00h.</p> <p>The index values used for other data object types is implementation specific and has no meaning defined by this specification.</p> <p>Undefined if Vendor ID value is FFFFh.</p> |
| 31:24 | <p>Next Index – Indicates the next DOE Discovery Index value. If the responding DOE instance supports entries with indices greater than the index indicated in the received DOE Discovery Request, it must increment the queried index by 1 and return the resulting value in this field.</p> <p>Must be 00h to indicate the final entry.</p> <p>Undefined if Vendor ID value is FFFFh.</p> |



Figure 6-45 DOE Discovery Response Data Object Contents (4th DW) §

Table 6-32 DOE Discovery Response Data Object Contents (4th DW) §

| Bit Location | Description |
|--------------|--|
| 17:0 | <p>Max DO Length – Maximum length supported for this data object type, expressed as a number of DWs including the 2 DWs of the Data Object Header, encoded such that a value of 0 0002h indicates 2 DWs, and a value of 0 0000h indicates 2^{18} DWs. A value of 0 0001h is not permitted.</p> <p>Undefined if Vendor ID value is FFFFh.</p> |
| 31:18 | <p>Additional Info – The contents of this field are Data Object Type specific. Values are defined for each Data Object Type in § Table 6-33 .</p> <p>Undefined if Vendor ID value is FFFFh.</p> |

PCI-SIG defined data object types are defined in § Table 6-33 .

Table 6-33 PCI-SIG Defined Data Object Types (Vendor ID = 0001h) §

| Vendor ID (h) | Data Object Type (h) | Description |
|---------------|----------------------|--|
| 0001 | 00 | <p>DOE Discovery – Every DOE instance must support this data object feature and associated type, both indicated by this entry.</p> <p>A requester uses DOE Discovery to discover all other Data Object types supported by the DOE instance.</p> <p>Must not include the <u>DOE Discovery Response Data Object Contents (4th DW)</u> .</p> |
| 0001 | 01 | CMA-SPDM – Data object type for SPDM messages. See § Section 6.31 . |

Base 6.4 vs Base 6.3

| Vendor ID (h) | Data Object Type (h) | Description |
|---------------|----------------------|---|
| | | Permitted to include the DOE Discovery Response Data Object Contents (4th DW) . The Additional Info field is Reserved. |
| 0001 | 02 | <p>Secured CMA-SPDM – Data object type for SPDM secure sessions. See § Section 6.31.4</p> <p>If this type is supported, then the CMA/SPDM (01h) type must also be supported.</p> <p>Permitted to include the DOE Discovery Response Data Object Contents (4th DW) . The Additional Info field is Reserved.</p> |
| 0001 | 03 | <p>CMA/SPDM with Connection ID – Data object type for SPDM messages with Connection ID.</p> <p>If this type is supported, then the CMA/SPDM (01h) type must also be supported.</p> <p>Required to include the DOE Discovery Response Data Object Contents 4th DW. Bits 11:0 of the Additional Info field must indicate the largest supported Connection ID value, such that all Connection IDs must be in the range from 0 to the indicated value:</p> <ul style="list-style-type: none"> 0000 0000 0000b 1 Connection ID ... 1111 1111 1111b 4096 Connection IDs <p>Bits 13:12 of the Additional Info field are Reserved.</p> |
| 0001 | 04 | <p>Secured CMA/SPDM with Connection ID – Data object type for SPDM secure sessions with Connection ID.</p> <p>If this type is supported, then the CMA/SPDM (01h) and CMA/SPDM with Connection ID (03h) types must also be supported.</p> <p>Required to include the DOE Discovery Response Data Object Contents 4th DW . Bits 11:0 of the Additional Info field must indicate the largest supported Connection ID value, such that all Connection IDs must be in the range from 0 to the indicated value:</p> <ul style="list-style-type: none"> 0000 0000 0000b 1 Connection ID ... 1111 1111 1111b 4096 Connection IDs <p>Bits 13:12 of the Additional Info field are Reserved.</p> |
| 0001 | 05 | <p>Async Message – Data object type for messages generated asynchronously by the device.</p> <p>Required to include the DOE Discovery Response Data Object Contents 4th DW .</p> |
| 0001 | 06 to FF | Reserved |

6.30.1.2 DOE Async Message §

The DOE Async Message data object feature allows a DOE instance to transfer device-initiated messages with system software, enabling the device to act as a requester in cases where otherwise the host would act as requester. The DOE Async Message data object type consists of 1 DW that indicates the contents, defined in § [Table 6-34](#), which, for Async

Message REQUEST (01h) or Async Message RESPONSE (02h), must be followed by an encapsulated DOE data object. The encapsulated data object must be a full data object as illustrated in § Figure 6-40.

Table 6-34 DOE Async Message Data Object Contents (1 DW) §

| ↓↑Bit Location↓ ↑↑Bit Location↑ | Description | | | | | | | | | | |
|------------------------------------|---|------------|--------------------------|------------|------------------------------|------------|-------------------------------|------------|---------------------------|---------------|----------|
| 7:0 | <p>Variety – Indicates Async Message variety:</p> <table style="margin-left: 20px;"> <tr><td>00h</td><td>Async Message GET</td></tr> <tr><td>01h</td><td>Async Message REQUEST</td></tr> <tr><td>02h</td><td>Async Message RESPONSE</td></tr> <tr><td>03h</td><td>Async Message DONE</td></tr> <tr><td>04-FFh</td><td>Reserved</td></tr> </table> | 00h | Async Message GET | 01h | Async Message REQUEST | 02h | Async Message RESPONSE | 03h | Async Message DONE | 04-FFh | Reserved |
| 00h | Async Message GET | | | | | | | | | | |
| 01h | Async Message REQUEST | | | | | | | | | | |
| 02h | Async Message RESPONSE | | | | | | | | | | |
| 03h | Async Message DONE | | | | | | | | | | |
| 04-FFh | Reserved | | | | | | | | | | |
| 31:8 | <p>Reserved – Requesters must place 00 0000h in this field. Responders must ignore the value in this field.</p> | | | | | | | | | | |

Each data object type definition must specify if that type is permitted to use the Async Message mechanism.

The following data object types are permitted in the encapsulated data object:

- CMA/SPDM
- Secure CMA/SPDM
- CMA/SPDM with Connection ID
- Secure CMA/SPDM with Connection ID

An Async Message is not permitted in the encapsulated data object.

A DOE instance advertises support for sending DOE Async Messages by having the DOE Async Message Support bit Set. A DOE instance is only allowed to send DOE Async Messages, when the DOE Async Message Enable bit is Set.

For a DOE instance to send a DOE Async Message to the host, it must Set DOE Async Message Status and wait to receive a DOE Async Message GET from the host.

System software supporting DOE Async Message must check the DOE Async Message Status bit, and, when that bit is Set and the system software is ready to process the message, then, the host must send an Async Message GET . The DOE instance must in response return the Async Message REQUEST . System software then, in turn, must send Async Message RESPONSE . If the DOE instance has more Async Messages to transfer, then the device must return another Async Message REQUEST , to which system software must return a corresponding Async Message RESPONSE . If the DOE instance has no more Async Messages to send, it must return Async Message DONE and Clear DOE Async Message Status .

A DOE instance must only return Async Message REQUEST in response to a received Async Message GET . It is permitted but not required for system software to abort (as defined in § Section 6.30.1) other data object transfers to give priority to Async Message transfers.

6.30.2 Operation §

Other than the DOE Discovery feature, DOE is used to transport data objects as part of a feature defined outside of DOE itself. This section defines requirements of DOE itself, although in some cases these requirements are permitted to be modified per the definition of a specific feature.

For features that define request/response protocols, unless there is a feature-specific requirement, a DOE instance must complete processing a received data object and, if a data object is required in response, must generate the response and Set the Data Object Ready bit in the DOE Status register within 1 second after the DOE Go bit was Set in the DOE Control Register , otherwise the DOE instance must Set the DOE Error bit in the DOE Status register within the same time limit. At any time, the system firmware/software is permitted to set the DOE Abort bit in the DOE Control Register , and the DOE instance must Clear the Data Object Ready bit, if not already Clear, and Clear the DOE Error bit, if already Set, in the DOE Status Register , within 1 second.

Once the transfer of a specific data object has been started, except in case of error or abort, that transfer must be completed before the transfer of another data object is started.

If a single DOE instance supports multiple data object features, system firmware/software is permitted to interleave data objects associated with different data object features, if and only if the data object features allow such interleaving.

Data object buffering requirements are determined by the data object feature(s) supported, and data object features must ensure that maximum data object sizes are well defined. When required, the Max DO Length must be used to indicate the maximum length supported for a specific data object type. All data object types that are defined to require the use of Capability Version 02h or greater in the Data Object Exchange Extended Capability must require the use of Max DO Length .

It is strongly recommended that implementations ensure that the functionality of the DOE Abort bit is resilient, including that DOE Abort functionality is maintained even in cases where device firmware is malfunctioning.

An FLR to a Function must result in the aborting of any DOE transfer in progress. Data object features must specify the handling of FLR and, as appropriate, other conditions that impact the data object feature. While an FLR is in-progress, all writes to the DOE Write Data Mailbox must be dropped, and all reads to the DOE Read Data Mailbox must be terminated with UR, or return all 0's. If these or other erroneous actions are detected while an FLR is in progress, it is permitted that, upon exit from FLR, the DOE Error bit be Set.

If software permits DOE operations to overlap with potential FLRs, software cannot depend on the RRS mechanism, but rather must use time-based mechanisms or FRS to determine the completion of an FLR.

It is not required that FLR result in any type of reset to the internal processing engine for DOE operations, although such behavior is permitted, and may be required or forbidden by specific data object features.

DOE errors cover errors in the operation of DOE itself, and, except as noted below, do not extend to errors associated with a data object feature. Any of the following events must result in the DOE Error bit being Set:

- A Poisoned Configuration Write to any of the DOE registers
- Overflow of the Write Data Mailbox mechanism
- Optionally, if the associated data object feature does not provide an alternate mechanism for reporting such errors, the transfer of a data object that is shorter than the minimum expected length of that data object

No response must be generated when the DOE Error bit is Set because of a condition associated with the receipt of a data object that would, in a non-error condition, have an associated data object response, no response must be generated.

Hardware behavior is undefined unless software issues writes to the DOE Write Data Mailbox Register and reads from the DOE Read Data Mailbox Register with all Bytes enabled. It is permitted, but not required, for hardware to check for accesses without all Bytes enabled, and if checked, it is strongly recommended that violations result in the DOE Error bit being Set.

The optional DOE Attention Mechanism supports improved power management of DOE implementations, by enabling a DOE instance to temporarily enter an unresponsive state, and providing software a mechanism to direct the instance back to a responsive state. If the DOE Attention Mechanism Support bit is Set, for backwards compatibility, the default is that the DOE instance must remain in a responsive state. System software is recommended to Set the DOE Attention Not Needed bit when it is acceptable for the DOE instance to enter and stay in a state where it is not immediately available for

use. When that bit is Clear, the DOE instance must remain in a state where it is ready to respond in a timely way to system software. The DOE At Attention bit, when Set, indicates the DOE interface is presently in a state of readiness. This bit must only be Cleared if the DOE Attention Not Needed bit in the DOE Control Register is Set. It is permitted for this bit to remain Clear for up to 50 ms following the Clearing of DOE Attention Not Needed.

IMPLEMENTATION NOTE: EXCHANGE OF DATA OBJECTS §

Exchange of Data Objects Data objects are exchanged through the mailboxes provided by the Data Object Exchange (DOE) Extended Capability. The DOE mailbox is defined to flexibly support a variety of data objects, and as a result of the definition of specific data objects and their associated features, it is necessary to provide the information required for requesters and responders to appropriately size their data buffers and robustly implement the associated feature.

At the level of individual DW transfers, the DOE responder can use the Completions for the DOE mailbox Configuration Read and Write operations as a flow control mechanism. It should be understood by hardware designers, however, that delaying Configuration Completions will typically stall the software operating DOE, and in some cases may also cause stalls in other software/hardware operations.

The DOE Busy bit can be used to indicate that the DOE responder is temporarily unable to accept a data object. It is necessary for a DOE requester to ensure that individual data object transfers are completed, and that a request/response contract is completed, for example using a mutex mechanism to block other conflicting traffic for cases where such conflicts are possible. The following example shows how system firmware/software transfers a request from system firmware/software to a DOE instance, and the response back to system firmware/software from the DOE instance:

1. System firmware/software checks that the DOE Busy bit is Clear to ensure that the DOE instance is ready to receive a DOE request.
2. System firmware/software writes the entire data object, starting with the first DWORD, a DWORD at a time via the DOE Write Data Mailbox Register.
3. System firmware/software writes 1b to the DOE Go bit.
4. The DOE instance consumes the DOE request from the DOE mailbox.
5. The DOE instance generates a DOE Response and Sets the Data Object Ready bit and generates a DOE Software notification, if supported and enabled.
6. System firmware/software waits for an interrupt if applicable, checks/polls the Data Object Ready bit and, provided it is Set, reads data from the DOE Read Data Mailbox Register and then writes any value to the DOE Read Data Mailbox Register to indicate a successful read, starting with the first DWORD, a DWORD at a time until the entire DOE Response is read.

The above example does not illustrate error handling or additional software mechanisms necessary to manage cases where more than one software entity could potentially attempt to use DOE.

6.30.3 Interrupt Generation §

A DOE instance is permitted to support the generation of DOE interrupts, as indicated by the DOE Interrupt Support bit in the DOE Capabilities Register. If DOE interrupts are supported, the DOE instance must support MSI and/or MSI-X. INTx interrupt signaling is not permitted with DOE. DOE interrupts are enabled by the DOE Interrupt Enable bit in the DOE Control Register. DOE interrupts are indicated by the DOE Interrupt Status bit in the DOE Status register.

If enabled (see § Section 6.1.4.3), an interrupt message must be triggered when the associated vector is unmasked (see § Section 6.1.4.5 Per-vector Masking and Function Masking), the DOE Interrupt Enable bit is Set, and the value of the DOE Interrupt Status bit transitions from 0b to 1b.

The interrupt message will use the vector indicated by the DOE Interrupt Message Number field in the DOE Capabilities Register . Multiple DOE instances in the same Function or RCRB are permitted to use the same interrupt vector.

6.31 Component Measurement and Authentication (CMA-SPDM) §

Component Measurement and Authentication/SPDM (CMA-SPDM) defines optional security features based on the adaptation of the data objects and underlying protocol defined in [SPDM]. These provide mechanisms to perform security exchanges (where this term is used generically to refer to all defined capabilities of [SPDM]) with a component, or Device/Function. It is intended that CMA-SPDM inherit all new capabilities of [SPDM] as that specification is enhanced, and identifiable by the versioning mechanisms defined for [SPDM]. CMA-SPDM is part of a layered architecture intended to support a consistent and structured approach to security (see § Figure 6-46).

As security requirements evolve, and sometimes older technologies become compromised, for example by new types of attacks that had not been foreseen, so that they are no longer considered secure, CMA-SPDM is intended to balance the need for maintaining sufficient security against the sometimes conflicting goal of maximizing interoperability. Backwards compatibility will be maintained wherever possible, but this may not be possible in all cases.

It must also be understood that many aspects of security, including the establishment of policies, and the evaluation of specific trust decisions, very often cannot be made locally to a specific device or platform, and so are necessarily outside the scope of this specification. CMA-SPDM should be understood as a tool that provides a common framework for some key elements of security.

Base 6.4 vs Base 6.3

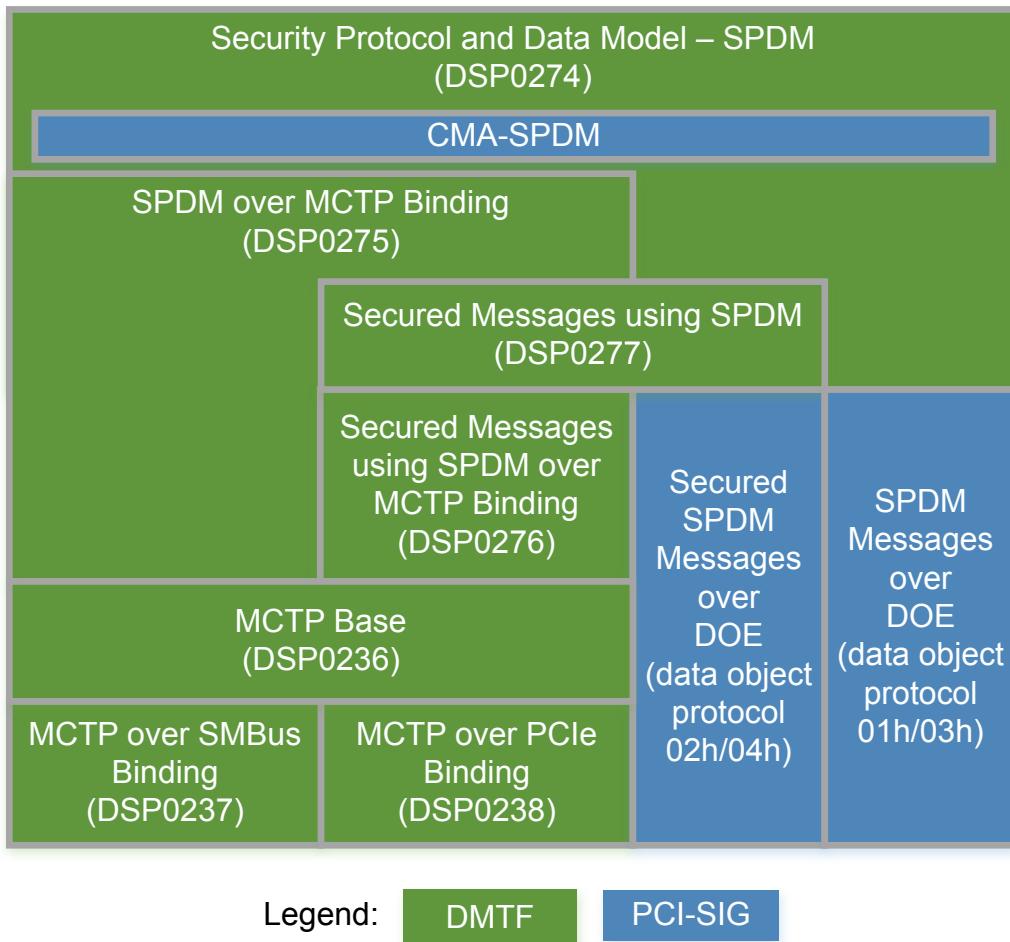


Figure 6-46 CMA-SPDM as Part of a Layered Architecture §

CMA-SPDM can use the Data Object Exchange (DOE) Capability (See § Section 6.30) as a mechanism for security exchanges, and can also be used for out-of-band or other in-band security exchanges, for example using MCTP [SPDM-MCTP] Binding.

It is possible to use CMA-SPDM in many scenarios—for example:

- Remote system administrators can dynamically generate a manifest of cryptographic identities of components of a system, especially at the level of removable units (e.g., add-in-cards/modules), without physical examination of the system, via a BMC or other platform root-of-trust.
- The identity of a Function can be verified by OS drivers before assigning resources to the Function during runtime/hot-plug scenarios without requiring a host reboot.
- Virtual Machine Monitors (VMMs) or TEE Security Managers (TSMs – see [§ Chapter 11](#)) can establish the hardware and firmware identities of a Function before assigning it to a Virtual Machine, and provide a service such to enable these to be confirmed by the Virtual Machine guest directly with the assigned Function.

The high-level overview of the CMA-SPDM security features and their associated PCIe-specific requirements are given in the following sections, whereas the foundational architecture, protocol and message definitions for the security exchanges used by CMA-SPDM are defined in [SPDM]. The messages exchanged between the requester and a responder for the CMA-SPDM security features are denoted as CMA-SPDM Messages.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: UNDERSTANDING AND IMPLEMENTING CMA-SPDM §

CMA-SPDM is part of a layered architecture to support device and platform security (see § Figure 6-46). Building on the DMTF Security Protocol & Data Model specification [SPDM], CMA-SPDM provides a “mapping” of that foundation, with the intent that future [SPDM] enhancements can, in most cases, be implemented without requiring modifications to CMA-SPDM. In addition to device authentication & firmware measurement, capabilities such as mutual authentication and cryptographic key exchange are also possible. Following [SPDM], CMA-SPDM uses the term “requester” to refer to agents initiating CMA-SPDM protocol requests, and “responder” to refer to an agent that ultimately services those requests and generates responses.

Depending on the use models supported, it may be desirable to implement support for more than one way of transporting CMA-SPDM requests and responses.

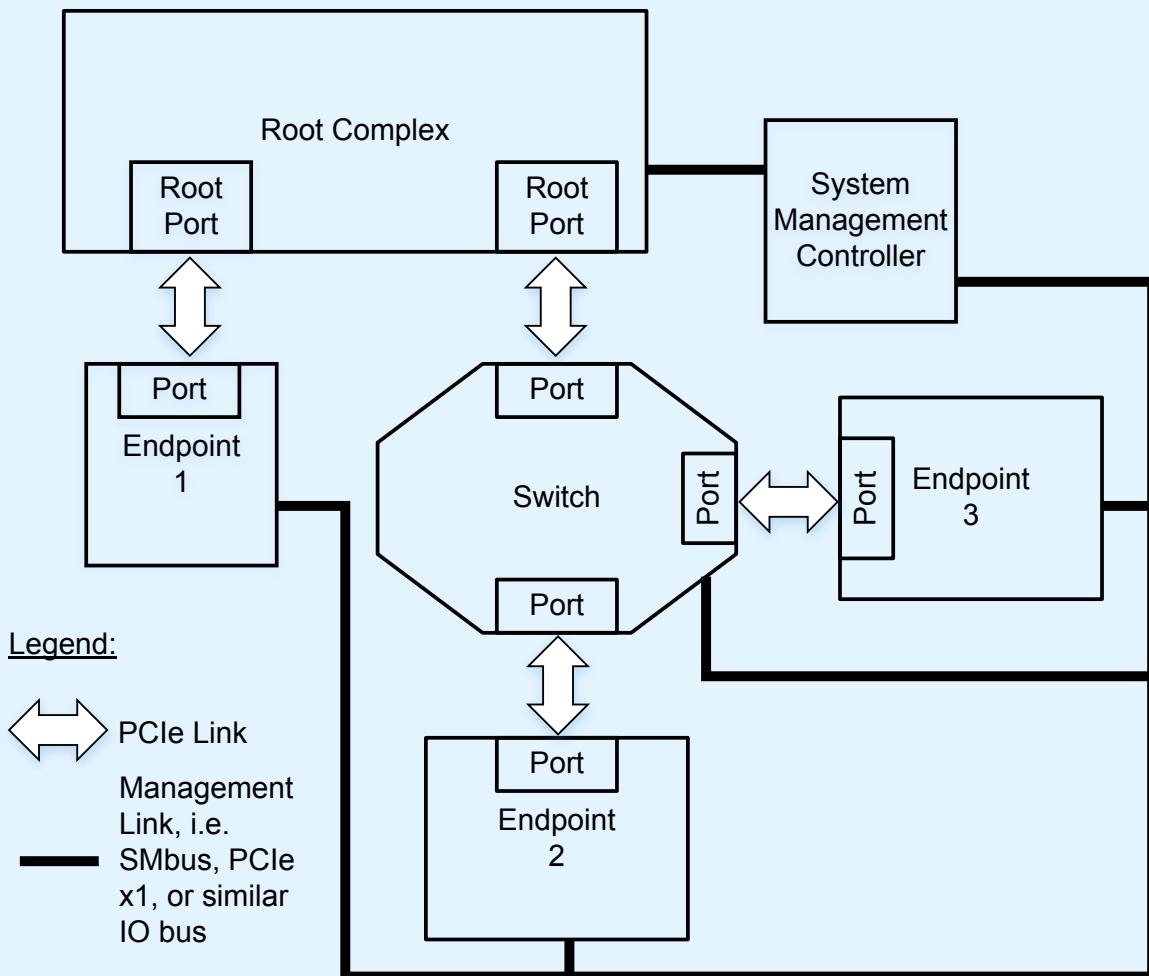


Figure 6-47 Example System Showing Multiple Access Mechanisms §

Base 6.4 vs Base 6.3

§ Figure 6-48 shows an example of a device that supports multiple platform use models and ~~multiple~~
~~multiple~~ access mechanisms. In this example, there is a System Management Controller that has an out-ofband connection to the other elements in the platform, enabling it to use CMA-SPDM even when the PCIe Links are not active and/or Fundamental Reset is active. When the PCIe Links are operating, CMA-SPDM can be used via [SPDM-MCTP]. The Root Complex can use CMA-SPDM over DOE. It is a platform implementation choice of which methods are used.

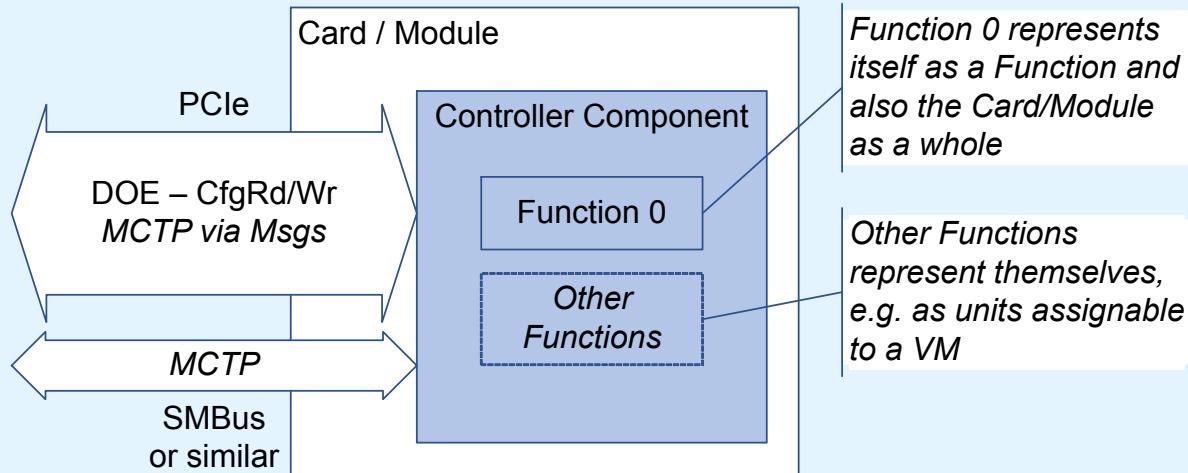


Figure 6-48 Example Add-In-Card Supporting CMA-SPDM §

Correspondingly, a device controller can support CMA-SPDM in multiple ways, depending on the associated form-factor and platform requirements and implementation choices. For maximum flexibility it may be desirable to support all of the following:

- Out-of-band (e.g., over SMBus, I2C, I3C, etc.) using MCTP, according to the specifications for [SPDM] + CMA-SPDM + [SPDM-MCTP]
- In-band using MCTP encapsulated in PCIe Messages, according to the specifications for [SPDM] + CMA-SPDM + [SPDM-MCTP] + [MCTP-VDM]
- In-band using the Configuration mailbox mechanism defined by the DOE Extended Capability , according to the specifications for [SPDM] + CMA-SPDM + DOE

Or, when the requirements for a specific implementation do not require this level of flexibility, it may be preferable to support only one or two of these approaches.

The platform requirements for CMA-SPDM are determined by the use cases supported. Example use cases that apply to an add-in-card as a unit include:

- Before platform boot, a management controller checks the add-in-card out of band.
- During platform boot, system firmware checks the add-in card in-band via DOE.
- During run-time, a management controller or system software/firmware checks the add-in-card via [MCTP-VDM].

To support use cases where the management controller and system firmware/software communicate the results of their checks with each other, it is necessary that the results of those checks be consistent, subject to any changes applied since earlier checks were made. It is for this use case that Function 0 is required to match identically the results received via these other mechanisms.

For use cases where the add-in-card is being evaluated as a unit, the device controller can, through implementation specific means, attest and/or measure other board elements, to ensure that the identity and integrity of the add-in-card as a whole is correct. How this can be done is outside the scope of CMA-SPDM.

For improved interoperability, CMA-SPDM requires support for certain algorithms, while allowing support for any of the algorithms supported by [SPDM]. Flexibility is allowed for Responder algorithm selection with the intent that device vendors can, if desired, align their choices with common standards such as those defined in the Commercial National Security Algorithm (CNSA) Suite and in NIST Special Publication 800-57.

The applications of CMA/SPDM are numerous and varied. For complex environments, the issues of secure identity provisioning, measurement collection/reporting, and verification of attested state may require evaluating additional standards and industry best practices. Examples include:

- TCG Reference Integrity Manifest (RIM) Information Model: <https://trustedcomputinggroup.org/resource/tcg-reference-integrity-manifest-rim-information-model>
- OCP Attestation of System Components v1.0 Requirements and Recommendations: <https://www.opencompute.org/documents/attestation-v1-0-20201104-pdf>

IMPLEMENTATION NOTE: OVERVIEW OF THREAT MODEL §

A detailed threat model analysis typically requires consideration of the context in which a system is operating and the composition of the system along with many other factors, and as such cannot be provided here. A high level overview of the types of threats for which CMA-SPDM may be applicable includes:

- Remote and Local software-based attacks, e.g., to install corrupted device firmware or roll back device firmware to an older version
- Threats from attacker in physical possession of the device including:
 - Software-based (similar to above)
 - Presentation (“impersonation”)
 - Hardware attacks of various sorts
- Attacks during manufacturing, provisioning or maintenance, including:
 - Provisioning of improper configuration and/or firmware
 - Improper “repair” of a module

CMA-SPDM requires the leaf certificate to include the information typically used by system software for device driver binding. This requirement is intended to support scenarios where an attacker device attempts to gain access to system resources by appearing to be a valid device type. Responding devices can include the device serial number value in the leaf certificate to simplify system implementation of policies that require specific unit instances to be identified, for example to support scenarios where a modified unit is substituted for a valid, but otherwise identical, unit.

6.31.1 Removed §

6.31.2 Removed §

6.31.3 CMA-SPDM Rules §

CMA-SPDM defines how the responder role as defined in [SPDM] must be implemented for PCIe devices, regardless of the communication path(s) implemented between the requester(s) and the responder.

It is permitted, but not required, to support CMA-SPDM and the responder role as defined in [SPDM] at the level of individual Functions. When this is done, then the Function(s) must implement both CMA-SPDM and DOE. For a Multi-Function Device, each Function implementing the responder role must implement CMA-SPDM and DOE. For Switches that support CMA-SPDM, each Switch [Upstream Port Function and Downstream Port](#) Function implementing the responder role must implement CMA-SPDM and DOE.

 Errata: Base 6.3
B823△◀▶

It is permitted, but not required, for CMA-SPDM to be implemented using one or more access mechanisms other than DOE, in support of various use models (e.g. see § [Figure 6-46](#)). When a use model requires CMA-SPDM to be applied at the level of a replaceable unit it may be necessary to support security exchanges operated by means other than DOE. For example, an add-in-card being evaluated by a Baseboard Management Controller (BMC), or similar element, may use MCTP over a sideband bus. For devices that implement such support, the certificate chain in slot 0 (referring to slots as defined in [SPDM]) must match identically the certificate chain in slot 0 returned by Function 0 via DOE, if DOE is also supported.

CMA-SPDM does not apply to Root Ports, and a Root Port must not implement CMA-SPDM.

The value of all measurements must always reflect the firmware in use at the time of the measurement being read, including for components that support runtime update of firmware without a system reset.

When using CMA-SPDM with DOE:

- The instance of DOE used for CMA-SPDM must support:
 - the DOE Discovery data object protocol,
 - the CMA-SPDM data object protocol,
 - if IDE is supported, the IDE_KM data object protocol using Secured CMA-SPDM (See § [Section 6.31.4](#)),
 - and no other data object protocol(s).
- A responder must support operation when the associated Function is in the D0 state, and is permitted but not required to support operation in non-D0 states.
- Behavior is undefined if a Function that does not support operation in non-D0 states is transitioned into a non-D0 state during a CMA-SPDM protocol operation.
 - It is strongly recommended that system software avoid placing a Function into a non-D0 state while a CMA-SPDM protocol operation is taking place.
- An FLR to a Function during the processing of a CMA-SPDM request must result in that Function terminating its processing of the request, and that Function not returning a response to the request.
 - An FLR to a Function during the DOE transfer of a CMA-SPDM request or response data object must follow the rules defined in § [Section 6.30](#).

When using CMA-SPDM with a transport mechanism other than DOE:

- A Responder must support operation whenever that transport is active, including cases where the device is in Conventional Reset, unless exceptions are allowed through means outside the scope of this specification
- CMA-SPDM context must be maintained by the Responder such that each transport mechanism (including DOE, when implemented) functions independently of all other transport mechanisms.
- All transport mechanisms must be treated as independent SPDM connections.
- When CMA-SPDM via DOE in Function 0 is also supported, the certificate chain from slot 0 retrieved via DOE must match the certificate chain from slot 0 retrieved via any/all other transport mechanisms.

For the CMA-SPDM data object type or SPDM with connection ID data object type, the SPDM message payloads must start from “Data Object DW 0”. The SPDM message payloads must follow [SPDM] specification Generic SPDM message field definitions, starting with SPDM version, Request Response Code, Param 1 and Param 2. The Byte mapping of SPDM Messages is shown in § Figure 6-49 . If required, SPDM Message payloads must be padded with 0’s to maintain DW alignment, when using DOE.

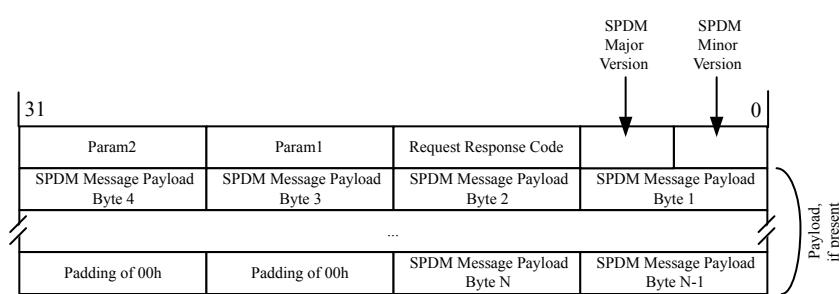


Figure 6-49 Byte Mapping of SPDM Messages Including Example Payload §

Without the connection ID, one DOE instance can only have one SPDM connection. With the connection ID, one DOE instance can have multiple SPDM connections. The maximum valid connection ID can be discovered by using DOE Discovery Data Object Protocol.

Some components provide a debug mode where a debugger is granted access to hardware security properties, allowing the debugger to influence the measurement process itself. It is strongly recommended that components report when a debug mode is active. The reporting mechanism is outside the scope of this specification. It can be SPDM debug and device mode in measurement record or DICE operation flags in DiceTcbInfo.

- For BaseAsymAlgo, Requesters must support all, and Responders must support one or more of the following:
 - TPM_ALG_RSASSA_3072
 - TPM_ALG_ECDSA_ECC_NIST_P256
 - TPM_ALG_ECDSA_ECC_NIST_P384
- For BaseHashAlgo, Requesters must support all, and Responders must support one or both of the following:
 - TPM_ALG_SHA_256
 - TPM_ALG_SHA_384
- For MeasurementSpecification, Requesters and Responders must support the following:
 - DMTF measurement specification format
- Requesters and responders must, for MeasurementHashAlgo, Requesters must support all, and Responders must support one or both of the following:

- TPM_ALG_SHA_256
- TPM_ALG_SHA_384
- It is permitted for Requesters and Responders to support additional algorithms defined for [SPDM] beyond those required.
- Responders must implement a Cryptographic Timeout (CT), as defined in [SPDM], of not more than 2^{23} µs.
 - Per [SPDM] CT is in turn indicated through the value of CTExponent.
 - It is strongly recommended that the CT be as short as practical.

6.31.4 Secured CMA-SPDM §

Secured CMA-SPDM provides security for data object protocols based on [SPDM] mechanisms, including the IDE key management (IDE_KM) protocol. Once a secure session has been established per [SPDM] (Revision 1.1 or later), it is permitted, and for some uses required, to use Secured CMA-SPDM to transfer other Data Objects with integrity/encryption, using the algorithm negotiated in the SPDM session establishment, per [Secured SPDM].

It is permitted to continue to perform non-secured CMA-SPDM operations after a session has been established, provided the specific use allows non-secure transport.

Secured CMA-SPDM data objects must be formatted per [Secure-SPDM]. It is permitted for specific data object protocols to constrain the use and content of optional fields, but if no such constraints are applied then the use of such fields is implementation specific.

An FLR to a Function for which there is an established secure session must not change the state of the secure session. However, as with non-secured CMA-SPDM, an FLR to a Function during the processing of a CMA-SPDM request must result in that Function terminating its processing of the request, and that Function not returning a response to the request, and this may impact the usability of the secure session and possibly render the secure session unusable.

For the Secured CMA-SPDM data object type or Secured CMA-SPDM with connection ID data object type, the Secured CMA-SPDM message payloads must start from “Data Object DW 0”. The Secured CMA-SPDM message payloads must follow the Secured CMA-SPDM specification Secure Message fields definition, starting with Session ID. “Sequence Number” field must be absent (S=0). “Random Data” field must be absent (R=0). The “Application Data” field must be the in-session SPDM message. The Byte mapping of Secured CMA-SPDM Messages is shown in § Figure 6-50 . If required, Secured CMA-SPDM Message payloads must be padded with 0’s to maintain DW alignment, when using DOE.

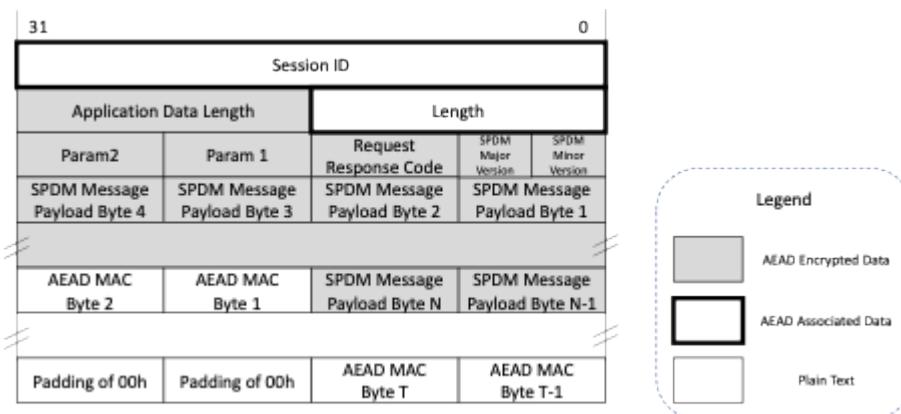


Figure 6-50 Byte Mapping of Secured CMA-SPDM Messages Including Example Payload §

Additional Rules:

- Requesters and Responders must support ENCRYPT_CAP and MAC_CAP.
- For DHE group, Requesters must support all, and Responders must support one or both of the following:
 - secp256r1
 - secp384r1
- For AEAD structure, Requesters and Responders must support the following:
 - AES-256-GCM with 16 Byte MAC
- For KeySchedule structure, Requesters and Responders must support the following:
 - SPDM Key Schedule
- For Secured Message format, Requesters and Responders must support the following:
 - DSP0277 Secured Messages opaque data format as defined in Section 8 of DSP0277, if SPDM 1.1 or below is used.
 - DSP0274 General opaque data format as defined in Section 14 of SPDM 1.2, if SPDM 1.2 or above is used. The OtherParamsSupport field of NEGOTIATE_ALGORITHMS request and ALGORITHMS response must set the bit for OpaqueDataFmt1.
- Requesters and Responders must use DSP0277 specification version as the Secured Message transport binding version (format as defined in Section 6 of DSP0277) required in the OpaqueData fields of KEY_EXCHANGE request / KEY_EXCHANGE_RSP response / PSK_EXCHANGE request / PSK_EXCHANGE_RSP response messages defined in DSP0274, when using DOE.
- Requesters and Responders may support mutual authentication. If mutual authentication is supported, for ReqBaseAsymAlgo, Requesters and Responders must support one or more of the following:
 - TPM_ALG_RSASSA_3072
 - TPM_ALG_ECDSA_ECC_NIST_P256
 - TPM_ALG_ECDSA_ECC_NIST_P384
- Requesters and Responders are permitted to implement additional algorithms defined in [SPDM]

6.32 Deferrable Memory Write §

The Deferrable Memory Write (DMWr) is an Optional Non-Posted Request that enables a scalable high-performance mechanism to implement shared work queues and similar capabilities. With DMWr, devices can have a single shared work queue and accept work items from multiple non-cooperating software agents in a non-blocking way. A DMWr Completer, i.e., a Function whose DMWr Completer Supported bit is Set, is a Function that supports being the target of DMWr Requests.

The mechanisms for generating DMWr Requests are outside the scope of this document. For cases where a DMWr Requester supports generation of DMWr Requests directly via software mechanisms, it is recommended that the software thread that invoked the Request is informed of the corresponding Completion Status in an implementation specific manner.

As with AtomicOps (See § Section 6.15), the use model for DMWr requires that, from the initial trigger to generate a DMWr Request, the subsequent routing to the Completer, the Completer's resulting actions, and the return of the corresponding Completion, that each of these be “atomic” in the sense that a single trigger must result in a single Request, which must not be subdivided, being acted upon by the Completer in such a way that no observer can see a partial result, and a single Completion being returned to the Requester, without subdivision. Therefore, the following rules apply:

- Switches and Root Complexes that support DMWr routing must route DMWr Requests and Completions without modification.
- DMWr Completers must ensure that operations performed on behalf of a given DMWr Request are performed atomically with respect to each host processor or device access to that target location range.
- The internal implementation of DMWr Requesters and Completers to enforce the required “atomic” behavior is outside the scope of this document.

The target of a DMWr Request that is not a DMWr Completer (i.e., a Function whose DMWr Completer Supported bit is Clear) responds as per the Request handling rule for a Request Type that is not supported (see § Section 2.3.1). In Non-Flit Mode, components designed to earlier revisions of this specification that treat the Request Type value corresponding to DMWr Request as Reserved are permitted to treat a received DMWr Request as a Malformed TLP.

The following requirements apply to DMWr Completers:

- Functions indicate their ability to act as DMWr Completers by Setting DMWr Completer Supported and by indicating the largest DMWr TLP that the Function can receive (see DMWr Lengths Supported).
- Properly formed DMWr Requests must be handled as a Successful Completion (SC), Request Retry Status (RRS), Unsupported Request (UR), or Completer Abort (CA).
 - Properly formed DMWr Requests with types or operand sizes not supported by the Completer, or targeting an address not intended by the device’s programming model to be a target of DMWr Requests, or crossing an address boundary between two different resources, must be handled as Completer Abort (CA), and the value of the target location must remain unchanged.
 - Switches that support DMWr Routing but do not support serving as a DMWr Completer must handle properly formed DMWr Requests that target internal resources of the Switch as Completer Abort (CA), and the value of the target location must remain unchanged.
- When a DMWr Request cannot be completed successfully due to a temporary condition, the Completer is permitted to return a Completion with Request Retry Status (RRS)
 - When this is done, the value of the target location must remain unchanged, and the Completer must not assume that the Requester will repeat the request.
 - This is not an error
- Completers are permitted to use implementation specific mechanisms to determine when to use the RRS Completion Status in order to establish policies for fairness or for other reasons, and these mechanisms may be based on Requester ID, Traffic Class, PASID, payload contents, or other criteria as may be appropriate.
- Completers supporting DMWr are allowed to implement a restricted programming model.
 - If a Request that is not a DMWr targets an address intended by the device’s programming model to be a target of DMWr Requests, it is strongly recommended that the value of the target location remain unchanged, and, if the Request is a Non-Posted [Request or a UIO](#) Request, that the Completion returned does not include any sensitive information.
 - See Implementation Note: Optimizations Based on a Restricted Programming Model in § Section 2.3.1 for additional general guidance.
- Completers supporting DMWr that return Successful Completion (SC) must guarantee that the observed update granularity will not be smaller than 64 bytes, or the size of the Request, whichever is smaller.
 - This requirement applies to DMWr targeting “plain” memory, a shared work queue, or other implementation specific structures, when such operations are supported by the programming model of the Completer.
 - See also § Section 2.4.3 and § Section 2.4.4 .

Errata: Base 6.3
B834△↔

- If any Function in a Multi-Function Device associated with an Upstream Port supports DMWr Completer or DMWr routing capability, all Functions with Memory Space BARs in that device must decode properly formed DMWr Requests and handle any they don't support as an Unsupported Request (UR).
 - In such devices, Functions lacking DMWr Completer capability are forbidden from handling properly formed DMWr Requests as Malformed TLPs.
- Unless there is a higher precedence error, a DMWr-aware Completer must handle a Poisoned DMWr Request as a Poisoned TLP Received error (see § Section 2.7.2.1).
 - The Completer must return a Completion with a Completion Status of either Unsupported Request (UR) or Request Retry Status (RRS).
 - The value of the target location must remain unchanged.
- If the Completer of a DMWr Request encounters an uncorrectable error accessing the target location, the Completer must handle it as a Completer Abort (CA).
 - The subsequent state of the target location is implementation specific.
- Completers are permitted to support DMWr Requests on a subset of their target Memory Space as needed by their programming model (see § Section 2.3.1).
 - Memory Space structures defined or inherited by PCI Express (e.g., the MSI-X Table structure) are not required to be supported as DMWr targets unless explicitly stated in the description of the structure.
- If an RC has any Root Ports that support DMWr routing capability, all RCiEPs in the RC reachable by forwarded DMWr Requests must decode properly formed DMWr Requests and handle any they do not support as an Unsupported Request (UR).

The following requirements apply to Root Complexes and Switches that support DMWr routing:

- Root Ports and Switch Ports indicate their support of DMWr routing by Setting DMWr Request Routing Supported and by indicating the largest DMWr TLP that the associated Function supports (see DMWr Lengths Supported).
- Switches and Root Ports supporting the DMWr routing capability or DMWr Completer capability (or both) that receive a properly formed DMWr Requests must either forward it to another Port or handle it as a Successful Completion (SC), Request Retry Status (RRS), Unsupported Request (UR), Completer Abort (CA), or DMWr Egress Blocked error.
- If a Switch supports DMWr routing for any of its Ports, it must do so for all of them.
- For Switches and Root Ports supporting the DMWr routing capability, if a DMWr Request is received that crosses a decoding boundary between two different destinations, the Ingress Port must not propagate the Request, and must return a Completion with a Completion Status of UR.
- For a Switch or an RC, when DMWr Egress Blocking is enabled in an Egress Port and a DMWr Request targets going out that Egress Port, then the Egress Port must handle the Request as a DMWr Egress Blocked error and must also return a Completion with a Completion Status of UR.
 - If the severity of the DMWr Egress Blocked error is non-fatal, then this case must be handled as an Advisory Non-Fatal Error as described in § Section 6.2.3.2.4.1.
 - This is a reported error associated with the Egress Port (see § Section 6.2).
- For an RC, support for peer-to-peer routing of DMWr Requests and Completions between Root Ports is optional and implementation specific.
 - When supported, the associated Ports must Set the DMWr Request Routing Supported in the Device Capabilities 3 Register .
 - When supported, DMWr TLPs must be routed without modifying the size of the data payload.
 - Even when DMWr Request Routing Supported is Set in two Root Ports, it is implementation specific whether forwarding is supported between those Ports.

- If one Root Port in a Root Complex supports DMWr Completer or DMWr Routing, a DMWr Request received by that Port that is routed to another Root Port that does not support DMWr must be handled as an Unsupported Request (UR).
 - This is a reported error associated with the Ingress Port (See § [Section 6.2](#)).

The following requirements apply to DMWr Requesters:

- A Function is only permitted to generate DMWr Requests when the DMWr Requester Enable bit in the Device Control 3 register is Set.
- When a DMWr Request is completed with Request Retry Status (RRS), the Requester is permitted, but not required, to re-issue the Request.
 - The Requester is permitted to use any implementation specific criteria to determine if/when to re-issue the Request.
 - Subsequent Requests are permitted to be the same or modified.

IMPLEMENTATION NOTE: CONSIDERATIONS FOR THE USE OF DEFERRABLE MEMORY WRITE (DMWR) §

The intended use model for the Deferrable Memory Write (DMWr) is to implement efficient hardware/software interface control mechanisms, using specialized hardware in the Completer to process the DMWr Request and generate the appropriate Completion Status. For example, a device could implement “enqueue registers” that enable commands to the device to be issued via a single DMWr Request, and based on the Completion Status the device can indicate to the Requester of the command was accepted.

Users of DMWr must understand the functional implications of transaction ordering. A DMWr Request is a Non-Posted Request with Data, which means that Posted Requests are permitted to pass DMWr Requests. Additionally, there is no guaranteed ordering among all types of Non-Posted Requests (see Table 2-4, entries B3, B4, C3 & C4).

As with all types of “control” mechanisms, it is necessary for all participants to comprehend the specific requirements placed by the particular mechanism, and these will vary between different systems and different device types. In many cases it will be necessary to distinguish Requests issued from different software environments (e.g., from multiple Virtual Machine guests where the guests use different drivers) all sharing the same work queue. PASID is one mechanism that can be used for this purpose, although there are many alternatives (e.g., different ranges of addresses could be assigned to each environment that would be mapped to the same resources in the Completer). In some systems, system and application level software is capable of generating DMWr Requests according to a specific template (§ Figure 6-51), where bits 31:0 are defined by the system architecture, the P bit at bit 31 indicates if user (0b) or supervisor (1b) code triggered the Request, and bits 19:0 of the payload include the PASID to indicate the context in which the Request was generated.



Figure 6-51 Example DMWr Data Payload Template §

For performance reasons it is not recommended that DMWr be used for sending bulk data.

Being a Non-Posted Request, DMWr TLPs require a Completion. In addition, PCIe ordering rules dictate that Non-Posted TLPs cannot pass Posted TLPs, making Posted transactions preferable for improved performance.

Because DMWr TLPs and Memory Read Request TLPs can pass each other, and DMWr TLPs can be deferred by the Completer, care must be taken by Device and Device Driver manufacturers when attempting to read a memory location that is also the target of an outstanding DMWr Transaction, if this is even supported by the programming model of the Completer. Because of these properties, use of DMWr TLPs when transferring large amounts of data is not recommended.

When DMWr Transactions are used to enable a shared work queue, care must be taken to ensure that no Requesters are denied access indefinitely to the queue due to competition with other Requesters. Software entities that submit work to such a queue may choose to implement a flow control mechanism or rely on a particular programming model to ensure that all entities are able to make forward progress, for example to

include a feedback mechanism or an indication from the Function to software on the state of the queue, or a timer that delays DMWr Requests after a Completion with Completion Status RRS. The DMWr mechanism does not itself provide protection against software entities issuing Requests (either maliciously or unintentionally) at a rate high enough to cause problems with other software entities accessing the single shared work queue. The details of such mechanisms and programming models are outside of the scope of this specification.

6.33 Integrity & Data Encryption (IDE) §

Integrity & Data Encryption (IDE) provides confidentiality, integrity, and replay protection for TLPs Transmitted and Received between two Ports. It flexibly supports a variety of use models, while providing broad interoperability. The cryptographic mechanisms are aligned to industry best practices and can be extended as security requirements evolve. The security model considers threats from physical attacks on Links, including cases where an adversary uses lab equipment, purpose-built interposers, malicious Extension Devices, etc. to examine data intended to be confidential, modify TLP contents, reorder and/or delete TLPs. TLPs can be secured as they transit Switches, extending the security model to address attacks mounted by reprogramming Switch routing mechanisms, or using “malicious” Switches. IDE can be used to secure traffic within trusted execution environments, also known as trust domains, composed of multiple components – the frameworks for such composition are outside the scope of IDE.

IDE establishes an IDE Stream between two Ports (see § Figure 6-52). There are two types of IDE Streams: a Selective IDE Stream applies to selected TLPs as determined by association rules defined in this section, and a Link IDE Stream applies to all TLPs Transmitted using a particular TC except those that are associated with a Selective IDE Stream. When there are no Switches between the Ports, then it is possible to secure all, or only selected, TLP traffic on the Link, using Link IDE or Selective IDE, respectively. There is no required relationship, or restriction, between Link IDE and Selective IDE. It is possible to use both Link IDE and Selective IDE between two directly connected Ports, as shown in § Figure 6-52 between Ports A and B, in which case TLPs associated with the Selective IDE Stream are secured using that Stream’s key set, and all other TLPs are secured using the key set for the Link IDE Stream. Such a configuration may be desirable if, for example, different security policies are applied to the Selective IDE TLPs than to other Link traffic. It is possible to use Selective IDE in cases where the IDE Terminus is a Switch Port, as shown between Ports C and D. IDE does not establish security beyond the boundary of the two terminal Ports, and mechanisms for securing and/or isolating secure traffic within a Component are outside the scope of this document. Again referring to the example shown in § Figure 6-52, the Selective IDE Streams between Ports C and G, and between Ports G and H, are secured as they pass through the Switch. All other Link IDE and Selective IDE streams illustrated are secured by IDE from Port to Port, but must be secured by implementation specific means within the Component past the terminal Port. By implication, when Link IDE is used with TLPs flowing “hop-by-hop” through one or more Switches, it is necessary to ensure acceptable security is maintained within the Switch(es), but how this is done is outside the scope of this document.

Base 6.4 vs Base 6.3

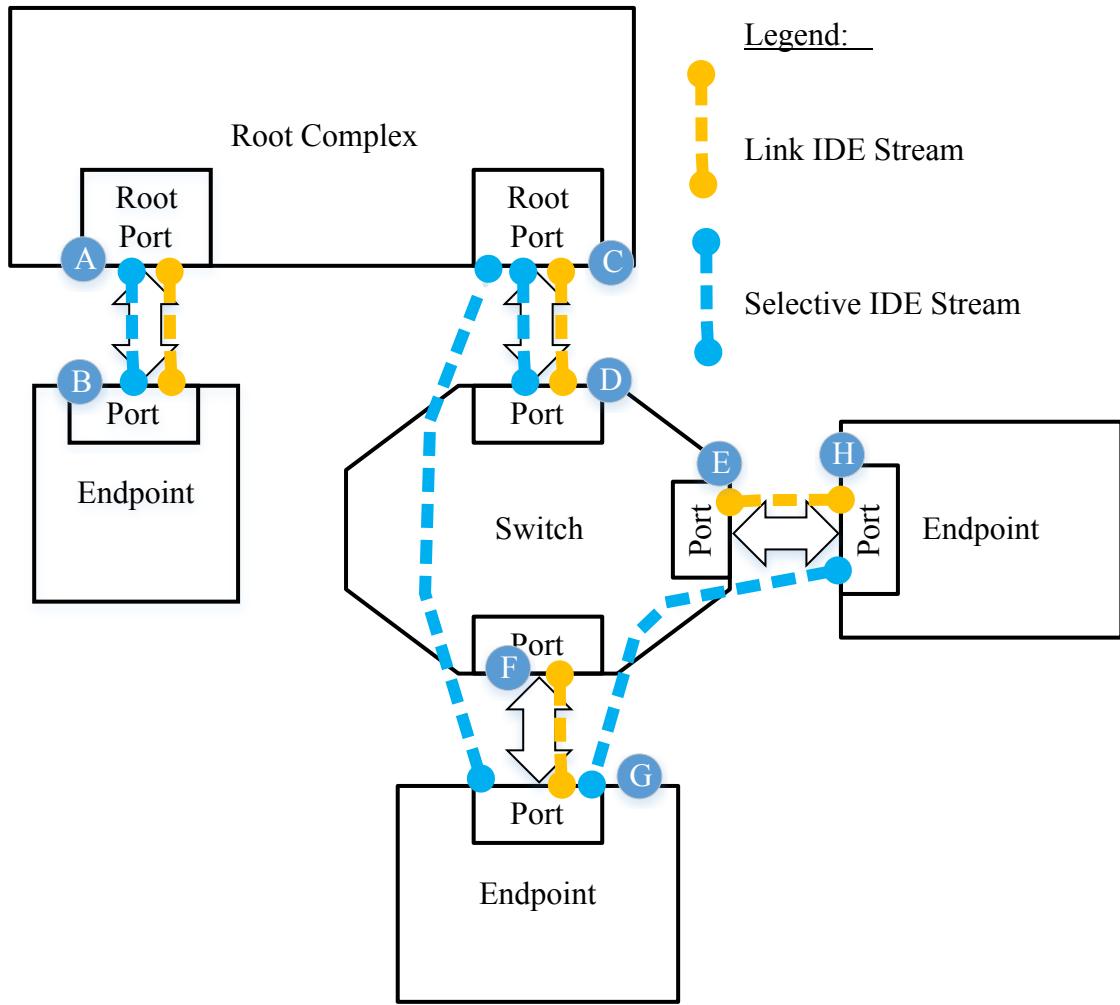


Figure 6-52 IDE Secures TLPs Between Ports §

In addition to the in-line securing of TLPs, as a “data plane” capability, IDE defines interoperable mechanisms for establishing Streams and programming keys, as a “control plane” capability, based on industry specifications. For example, for an Endpoint connected directly to the Root Complex (A to B above), one way to establish IDE is to use IDE Key Management (IDE_KM – see § Section 6.33.3) via DOE to allow host Firmware/Software to configure the Ports, including securely programming the IDE keys into both Ports. In another example, for two Endpoints communicating peer to peer (G to H above), the two Endpoints can implement communication directly via [MCTP-VDM] and [Secured-MCTP], where one will take the Requester role and the other the Responder role, and then applying the IDE_KM flow for secure key establishment. In an alternate example, it is also possible for some kind of management controller to apply IDE_KM over a sideband management connection, to program IDE keys in Ports throughout a system. The mechanisms for a management controller to program keys into a Root Complex are outside the scope of this document.

Policies for establishing trust between elements in a platform are outside the scope of IDE. It is strongly recommended that a platform-appropriate policy be implemented via platform-specific means. It is not necessary that this policy be applied prior to establishing IDE Streams, and in some cases it may be preferable to establish IDE first, and subsequently

apply security policy mechanisms, or to apply some policy mechanisms prior to establishing IDE and some additional mechanisms after.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: THREAT MODEL AND RELATED CONSIDERATIONS FOR IDE §

This implementation note provides a very general treatment of the threat model assumed for IDE. A detailed treatment will necessarily require knowledge of the platform environment and other elements that are outside the scope of this document.

This threat model covers:

Attacks using a logic analyzer or interposer type device, including e.g. “rogue” Retimers, where the attack devices attempt to add or delete TLPs, observe TLP payload data, and/or reorder TLPs. Example attacks include delaying a flag write to bypass a data write causing stale data to be accepted, or delaying a read to bypass a write to same location causing a stale value to be read. Reordering is discussed in more detail in the Implementation Note "Detection of Improper Reordering".

IDE secures host systems against device substitution attacks because, once authenticated key exchange is completed, a subsequent attack by a different unit trying to masquerade as the authenticated unit will fail because the masquerading unit cannot generate IDE TLPs using the correct key.

Provided an implementation includes appropriate self-protection measures, IDE also supports the detection of attacks involving the removal of a unit, for example by moving the unit to a different system and attempting to operate the unit masquerading as the authenticated host.

It is assumed that in an attack, the attacker will thwart error reporting attempts, e.g. by blocking Messages from the Port that detected the error, and such reporting messages are only intended to be used in debugging improperly configured systems. If a specific use model requires timely detection of security failures, some type of “heartbeat” mechanism should be used, rather than assuming that the failure will be reported directly.

This threat model does not cover:

Security exposures caused by inadequate Device implementation. For example, implementations are necessarily required to secure local keys, interconnects, and memory, including, for example, local memory implemented on an add-in-card using discrete memory components. IDE does not protect against on-die traffic redirection, for example between Functions of a Multi-Function Device.

Debug mechanisms should be given careful review as they can easily cause information exposure when improperly implemented. It is strongly recommended that debug state be reported using measurement mechanisms, and that any change in debug configuration that could expose data intended to be secured result in a transition to Insecure (see § Section 6.33.1).

There are many considerations regarding secure key generation, programming, and storage, and it is strongly encouraged that non-experts consult with experts to evaluate all levels of implementation to ensure that good practices are followed. In all cases, it is essential to avoid exposure of plain text keys by any means including debug features such as tracers, configuration registers etc.

If partial header encryption (see § Section 6.33.4) is not used, “side channel” attacks may be possible in some cases based on attacker analysis of the information included in the headers. For example, see <https://www.ieee-security.org/TC/SP2015/papers-archived/6949a640.pdf>.

Considerations:

IDE secures TLP traffic from one Port to another Port. TLP content is not secured on-die by IDE past the terminal Partner Ports, and so it is necessary to provide appropriate implementation specific protective measures based on use model requirements to ensure that TLP traffic is secured prior to transmission, and following reception.

IDE assumes the implementation of appropriate isolation mechanisms to ensure that information remains secured beyond the Port to Port connection secured via IDE. In some cases, entire components can be considered “secure” and there is no need to distinguish traffic on-die, in other cases the establishment of one or more Trusted Execution Environments (TEEs) may be needed to isolate secure traffic from non-secure traffic, and different secure environments from each other. Although it is permitted to establish more than one IDE Stream between the same two Ports, this is not generally needed or useful, because it is assumed that once on-die, all secure traffic is “equally” secure, and using separate IDE Streams provides no additional protection. The details of how such TEEs are implemented and managed are outside the scope of IDE. The **XT bit** ECN: Base 6.3 XT $\triangleleft\triangleright$ and the **T bit** $\downarrow\downarrow\downarrow\downarrow$ **are** used for TEE management mechanisms (see § Chapter 11.). IDE mechanisms ensure that the **XT and** **T** **bit** **bits** (like other TLP content) $\downarrow\downarrow\downarrow\downarrow$ **are** secured during transit.

Good practices for implementing TEEs include, but are not limited to:

- Securing secrets through the use of local encryption, access control, and/or other mechanisms
- Ensuring that secure data cannot “leak” due to errors, power management, or other operations
- Detecting inappropriate attempts to reconfigure IDE, e.g. writes to any of the IDE control registers, and/or other internal conditions that could compromise secure data and taking appropriate measures, including potentially forcing the Port into Insecure
- Ensuring that secret keys are never exposed or stored in non-secure buffers
- Ensuring that the establishment & management of TEEs is itself secure

The implementation of TEEs can be very complex, and it is strongly recommended that persons with appropriate security expertise are intimately involved in the development and validation of components and systems.

Although Link IDE applies to all kinds of TLPs, Selective IDE can only be applied to certain types of TLPs (see § Table 6-35). Memory operations are required to be supported for virtually all use models, and are supported by Selective IDE. I/O operations are not commonly used, and are not supported by Selective IDE to simplify design and validation. Selective IDE can be applied to Messages, and, optionally, to Configuration Requests & Completions. Selective IDE can be applied to TLPs with Prefixes, but Local TLP Prefixes are not protected when the TLP is associated with a Selective IDE Stream. In NFM, End-End TLP Prefixes are protected along with the associated TLP, and in FM, OHC content is protected along with other Header content.

6.33.1 IDE Stream and TEE State Machines §

Conceptually, the initialization of a Link or Selective IDE Stream involves multiple steps, although some of these steps can be merged or performed in a different order. The first step is to establish the authenticity and identity of the components containing the two Partner Ports to be the IDE Terminuses of the IDE Stream. This may be done using CMA-SPDM, by implementation specific means, or in some cases implicitly. The second step is to establish the IDE Stream keys – the IDE Key Management (IDE_KM – § Section 6.33.3) provides a way to do this. Third, the Secure Connection must be configured, and, finally, the establishment of the IDE Stream is triggered.

Conceptually, each IDE Stream is associated at each Partner Port with a state machine illustrated in § Figure 6-53.

Base 6.4 vs Base 6.3

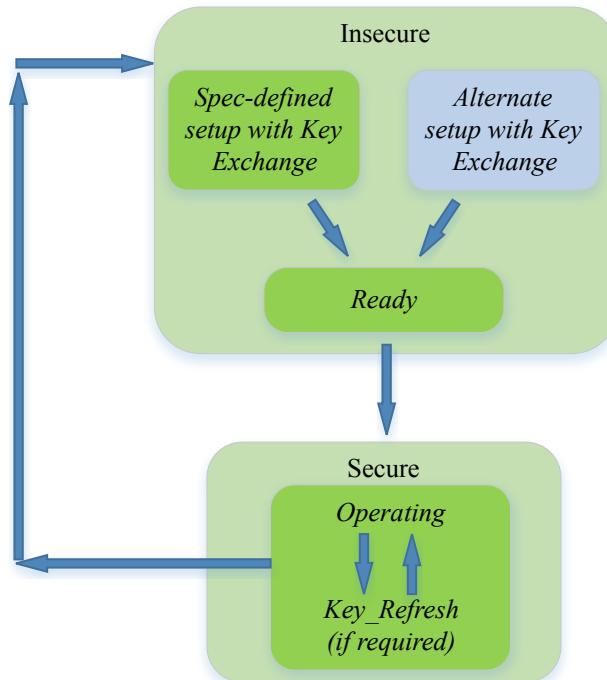


Figure 6-53 IDE Stream State Machine §

- The Insecure state indicates that the necessary steps to operate the IDE Stream have not been completed, or that some event has ended the operation of a previously operating IDE Stream.
 - Typically the Insecure will include various conceptual substates that are not directly observable by the hardware, and only the system firmware/software configuring the IDE Stream will have the ability to comprehend when all necessary steps have been completed.
 - The Ready conceptual sub-state of the Insecure state is entered when all necessary configuration has been performed; this condition must be tracked by system firmware/software.
 - In many cases it will not ~~not~~ be possible for hardware to distinguish when all necessary configuration has been performed, and there is no requirement for hardware to track the transition into the Ready sub-state.
- The IDE Stream State Machine for a specific Stream of a Port must transition from Secure to Insecure when the corresponding ~~Link/Selective IDE Stream Enable~~ Link/ Selective IDE Stream Enable bit is Cleared.
 - As further discussed below, it is essential that the Port internally block all transmissions that are intended to be secure if the corresponding IDE Stream State Machine is not in the Secure state.
- If at any time a condition compromising the security of the IDE Stream is detected at the Port, the Port must transition to Insecure.
 - It is permitted to transition to Insecure for implementation specific reasons.

A trusted execution environment (TEE) using IDE must prevent the transmission of TLPs intended to be secure using non-IDE TLPs, and must reject non-IDE TLPs received if the TEE requires those TLPs to be secure. A specific architecture for devices that support TEEs is defined by TDISP (see § Chapter 11.). In order to precisely define the normative requirements for IDE in relation to TEEs, we will assume that TEEs have internal states that correspond to those shown in § Figure 6-54.

Base 6.4 vs Base 6.3

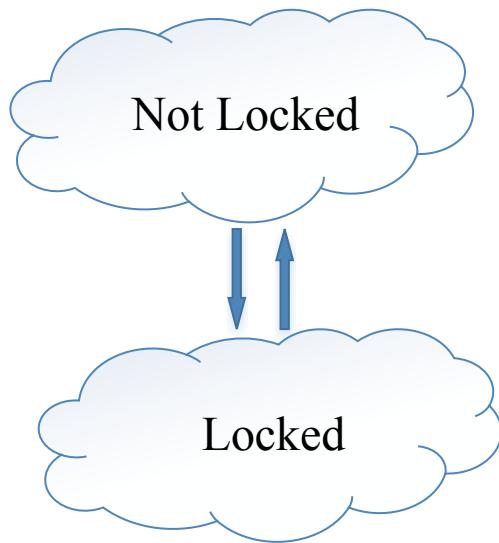


Figure 6-54 IDE Stream State Machine §

TDISP (see § Chapter 11.) defines specific requirements that extend the following rules, that apply broadly to all TEEs using IDE:

- A TEE must distinguish between at least two operating conditions, that may be called by any names but here will be called Not Locked and Locked.
 - A TEE must transition to the Locked state if and only if all IDE Streams required for the secure operation of the TEE are in the Secure state.
 - A TEE must transition to the Not Locked state if any IDE Stream required for the secure operation of the TEE is in the Insecure state.
- A TEE using an IDE Stream must precisely define the essential configuration information that could affect the security of the IDE Stream, and, once that IDE Stream is established, that essential configuration information must be confirmed and maintained by secure means so as to detect “adversary-in-the-middle” (AIM) attacks attempted during or after IDE Stream establishment, and changes to that information blocked and/or detected, as to detect/prevent attacks during operation.
 - The specific configuration information to which this requirement applies is implementation specific and dependent on the hardware elements involved, the security attributes required, and potentially on assumptions of use, all of which are outside the scope of this specification.
 - How the information is confirmed is implementation specific, but would typically include securely transferring a data structure that contains a local snapshot of the information to a secure partner to be compared against the expected values.

6.33.2 IDE Stream Establishment §

To establish IDE Streams interoperably based on this specification, system firmware/software acts as a central authority to create and program keys into the two Partner Ports. The following rules apply:

- For Endpoints, including Functions of a Multi-Function Device , associated with an Upstream Port, only Function 0 must implement the IDE Extended Capability .
- For Switches, including cases where one or more Functions of a Multi-Function Device represent the Upstream Port of a Switch, the IDE Extended Capability must only be implemented in Function 0, and implemented such that Function 0 represents the Multi-Function Device as a whole.
- For a Downstream Port, the Bridge Function associated with the Port must implement the IDE Extended Capability .
- All Ports other than Root Ports must implement support for key management by means of the IDE key management (IDE_KM) protocol defined in § Section 6.33.3 .
 - For Switch and Root Ports it is permitted for one Port to provide the DOE and CMA-SPDM responder function on behalf of other Ports as defined in § Section 6.33.3 .
 - Root Ports are permitted to implement support for the IDE_KM protocol.

It is also permitted for systems to implement the IDE_KM protocol via MCTP (see § Section 6.33.3).

It is also permitted for system firmware/software to enable pass-through communications between the two Partner Ports, where one of the two takes the Requester Role and the other takes the Responder Role, implementing the IDE_KM protocol defined below directly between the two Partner Ports (as an example see the Selective IDE Stream between Ports G and H in § Figure 6-52). How this is discovered and enabled is outside the scope of this specification.

6.33.3 IDE Key Management (IDE_KM) §

IDE Key Management (IDE_KM) builds upon [SPDM] and [Secured-SPDM], and can be used over multiple transports (see § Figure 6-55).

Base 6.4 vs Base 6.3

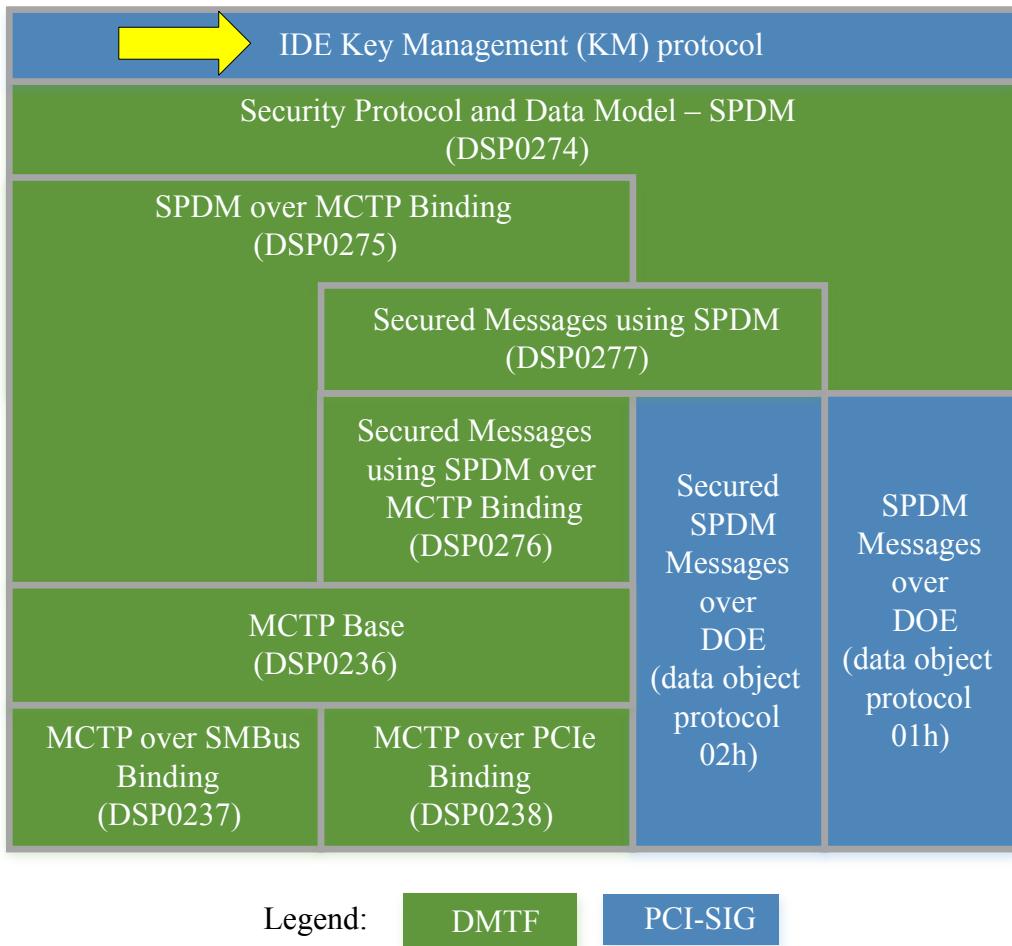


Figure 6-55 IDE Key Management (IDE_KM) and Related Specifications & Capabilities §

The following rules define the IDE key management (IDE_KM) protocol, and must be followed for Ports that support the use of IDE_KM:

- The IDE_KM protocol uses the data objects defined below, where:
 - The Requester must use the [SPDM] VENDOR_DEFINED_REQUEST format, the Responder must use the [SPDM] VENDOR_DEFINED_RESPONSE format.
 - The StandardID field of the VENDOR_DEFINED_REQUEST/ VENDOR_DEFINED_RESPONSE must contain the value assigned in [SPDM] to identify PCI-SIG.
 - The VendorID field of the VENDOR_DEFINED_REQUEST/ VENDOR_DEFINED_RESPONSE must contain the value 0001h, which is assigned to the PCI-SIG.
 - The VendorDefinedReqPayload/VendorDefinedRespPayload field of the VENDOR_DEFINED_REQUEST/ VENDOR_DEFINED_RESPONSE must be the data object content as defined below.
 - The VENDOR_DEFINED_REQUEST/ VENDOR_DEFINED_RESPONSE must in turn form the Application Data field of a Secured Message per [Secured-SPDM].

- It is strongly recommended that the cryptographic strength used in the secure session be at least as strong as selected for IDE itself.
- If any IDE_KM data object is received that has not been transferred securely per [Secured-SPDM], the received data object must not be used for key management, and, if it is a request, must not result in a response.
- The size of the VendorDefinedReqPayload/VendorDefinedRespPayload must match the size of the data object defined below.
 - If the size does not match, the received data object must not be used for key management, and, if it is a request, must not result in a response.
- For Endpoint Functions, including Functions of a Multi-Function Device , associated with an Upstream Port, Function 0 must implement DOE and CMA-SPDM for Authentication and key exchange, including the secure session establishment mechanism (see [SPDM]) and the IDE key management (IDE_KM) protocol as a Responder, as defined below.
 - IDE operates at a per-Port level, and Function 0 of an Upstream Port must be used for the purposes of establishing the authenticity and identity of the associated Component, performing key exchange, and the configuration and management of IDE Stream(s) for that Port.
- Each Upstream Port Function, regardless of Function Number, representing a Switch for which Link IDE Stream Supported and/or Selective IDE Streams Supported are Set, including in Multi-Function Devices , must implement DOE and CMA-SPDM supporting the IDE key management (IDE_KM) protocol as a Responder, as defined below.
- It is permitted for a Root Complex to:
 - support the IDE_KM protocol using a DOE instance for some or all Root Ports per-Port, or using some Root Ports to represent other Root Ports
 - implement a DOE instance in an RCRB supporting IDE_KM for Root Ports,
 - use implementation specific key management.
- For Switches and Root Complexes, it is permitted for one Port to implement the IDE_KM interface as a responder for itself and for other Ports of the Switch/Root Complex.
- Ports are permitted to support the IDE_KM protocol transported via MCTP.
- Within each data object, the Protocol ID field in bits [7:0] of the first DW must be 00h to indicate IDE.
- The Object ID field in bits [15:8] of the first DW indicates the IDE_KM data object type and is encoded as:
 - 00h: Query (QUERY)
 - 01h: Query Response (QUERY_RESP)
 - 02h: Key Programming (KEY_PROG)
 - 03h: Key Programming Acknowledgement (KP_ACK)
 - 04h: Key Set Go (K_SET_GO)
 - 05h: Key Set Stop (K_SET_STOP)
 - 06h: Key Set Go/Stop Acknowledgement (K_GOSTOP_ACK)
 - all other encodings Reserved.
- A Requester is permitted to use QUERY to determine the capabilities and configuration of a Port (see § Figure 6-56)
 - The PortIndex field must be used to indicate the Port addressed by the QUERY.
- A Responder must respond to a QUERY indicating a PortIndex value of 00h.

- IDE_KM assigns unique Port numbers (PortIndex) for each Port of an Endpoint, Switch or Root Complex, implementing IDE_KM.
- For a Switch that supports the IDE_KM responder role:
 - The Switch Upstream Port must implement the responder role for itself and the Upstream Port must respond to a PortIndex of 00h.
 - Downstream Ports of the Switch that are represented by the Upstream Port must respond to PortIndex ranging from 01h to FFh, where the order is established by the Device/Function numbers assigned by the Switch construction to the Downstream Ports from lowest to highest.
 - It is permitted for a Switch to implement a responder capability in a Downstream Port, for example by implementing a DOE instance in the Downstream Port, in which case that Downstream Port must respond to a PortIndex of 00h.
 - It is permitted for that Port to represent other Downstream Ports, in which case the represented Downstream Ports must respond to PortIndex ranging from 01h to FFh, where the order is established by the Device/Function numbers assigned by the Switch construction to the Downstream Ports from lowest to highest.
- For a Root Port that supports the IDE_KM Responder role:
 - The Root Port must implement the responder role for itself and must respond to a PortIndex of 00h.
 - It is permitted for that Port to represent other Root Ports, in which case the represented Root Ports must respond to PortIndex ranging from 01h to FFh, where the order is established by the Device/Function numbers assigned by the Root Complex construction to the Root Ports from lowest to highest.
- For an Endpoint Upstream Port that supports the IDE_KM responder role, the Port must respond to a PortIndex of 00h.
- Ports/RCRBs implementing the [SPDM] responder role must respond to a QUERY with QUERY_RESP (see § Figure 6-57).
 - The PortIndex field must contain the PortIndex field value from the corresponding QUERY.
 - The MaxPortIndex field value must indicate the maximum PortIndex value for the Ports represented by this Port/RCRB.
 - If only one Port is represented, including for all Endpoint Upstream Ports, the MaxPortIndex field must be 00h.
 - The Bus Number field must contain the Bus Number of the Function corresponding to the PortIndex field value.
 - The Segment field must:
 - for Ports that are not Root Ports, be zero
 - for Root Ports, contain the Segment Number value for the Root Port, or zero if the Root Complex implements only one Segment.
 - For Non-ARI Functions the Device/Function Number field must contain the Device and Function number of the Function corresponding to the PortIndex field value.
 - For ARI Functions the Function Number field must contain the Function number of the Function corresponding to the PortIndex field value.
 - The remainder of QUERY_RESPONSE must consist of the contents of the IDE Extended Capability Structure, other than the IDE Extended Capability Header itself, for the addressed Port.
 - The Supported Algorithms and Selected Algorithm field values returned in QUERY_RESP must be compared against the values read from the corresponding fields in the Responder Port's IDE Extended Capability structure and if non-matching values are detected then the

IDE_KM protocol for this secure session must be aborted, or other appropriate corrective action taken to avoid a possible “downgrade” attack.

- Requesters must not issue other requests after issuing a QUERY command until receipt of the corresponding QUERY_RESP.
- KEY_PROG, KP_ACK, K_SET_GO, K_SET_STOP and K_GOSTOP_ACK all apply to a single Sub-Stream, direction (Tx or Rx) and Key Set.
 - The Key Sub-Stream field indicates the Key Sub-Stream, using the same encodings as defined for the Sub-Stream identifier (see § [Section 6.33.5](#))
 - The direction is indicated by the RxTxB bit encoded:
 - 0b – Receive
 - 1b - Transmit
 - The Key Set field indicates the Key Set, corresponding to the K bit value in the IDE TLP Prefix (NFM)/OHC-C (FM).
- For Ports implementing the Responder role, key programming and the ability to select the initial value of the IV must be supported using the KEY_PROG command (see § [Figure 6-58](#)).
- The length of the key must correspond to the length indicated in the Selected Algorithm field for the Stream.
- The Requester must not send another KEY_PROG command to the same Port until it has received a KP_ACK from the Port.
 - If the Requester does not receive a KP_ACK from the Responder within 1 second plus a sufficient time to account for transport delay the Requester is permitted to consider that the Responder is not operating correctly.
- Fields specific to the KEY_PROG command are:
 - PortIndex, indicating the Port to which the key is to be programmed, corresponding to the order established in the QUERY_RESP
 - **Stream ID** **Stream_ID**
 - Key, which must be of the size required for the Selected Algorithm for the Stream
 - IFV, indicating the initial value for the invocation field of the IV, which must be 64 bits in size, and must initially set to the value 0000_0001h upon establishment of the Stream and when performing a key refresh.
- A Port implementing the Responder role must acknowledge receipt of a KEY_PROG command by returning KP_ACK, defined in § [Figure 6-59](#) .
 - The Status field must indicate the result of the KEY_PROG command, encoded as:
 - 00h: Successful
 - 01h: Failed to parse command – Incorrect Length
 - 02h: Failed to parse command – Unsupported value in PortIndex
 - 03h: Failed to parse command – Unsupported value in other field(s)
 - 04h: Unspecified Failure
 - 05-FFh: Reserved – Must not be used in generating KP_ACK, but if received must be treated as Unspecified Failure
 - The Responder must return KP_ACK within 1 second of the receipt of the KEY_PROG command.
 - It is strongly recommended to return KP_ACK as quickly as possible.
 - Return of KP_ACK, regardless of Status, indicates the Port is able to receive and process another KEY_PROG command .

- Mechanisms for generating keys are outside the scope of this document.
- It is strongly recommended to complete key programming for a Stream before Setting the Enable bit in the IDE Extended Capability entry for that Stream.
 - It is permitted, but strongly not recommended, to Set the Enable bit in the IDE Extended Capability entry for a Stream prior to the completion of key programming for that Stream.
- If the Enable bit is Set in the IDE Extended Capability entry for a Stream, but that IDE Stream is not already in Secure, the receipt of a K_SET_GO for must trigger the Port to Transmit/Receive IDE TLPs for the indicated Stream, Sub-Stream, direction and Key Set.
 - The agent implementing the Requester role for IDE_KM must send K_SET_GO commands to enable the Receivers at both IDE Partner Ports, and then send K_SET_GO commands to enable the Transmitters at both IDE Partner Ports.
 - The Port must use the indicated Key Set for IDE TLP transmissions associated with the IDE Stream, Sub-Stream, and direction starting not more than 10 ms after the receipt of the K_SET_GO command to enable the Transmitter.
 - The Port must be capable of processing received IDE TLPs using the indicated Key Set within 10 ms after the receipt of the K_SET_GO command enabling the Receiver.
 - The Port must respond by returning an K_GOSTOP_ACK once it is capable of receiving another IDE_KM Request.
- If the Enable bit is Set in the IDE Extended Capability entry for a Stream, and that IDE Stream is already in Secure (a key refresh operation), the receipt of a K_SET_GO for must trigger the Port to Transmit/Receive IDE TLPs for the indicated Stream, Sub-Stream, direction and Key Set.
 - The agent implementing the Requester role for IDE_KM must send K_SET_GO commands to enable the Receivers at both IDE Partner Ports, and then send K_SET_GO commands to enable the Transmitters at both IDE Partner Ports.
 - The Port must use the indicated Key Set for IDE TLP transmissions associated with the IDE Stream, Sub-Stream, and direction starting not more than 10 ms after the receipt of the K_SET_GO command to enable the Transmitter.
 - The Port must be capable of processing received IDE TLPs using the indicated Key Set within 10 ms after the receipt of the K_SET_GO command enabling the Receiver.
 - For each Sub-Stream of the Stream, until the Port receives an IDE TLP using the new key set, as indicated by the K bit value toggling, the Port must continue to accept IDE TLPs using the established key set.
 - Once the Port receives an IDE TLP using the new key set on a Sub-Stream it must invalidate and render unreadable the old key set, and discard subsequently received IDE TLPs using the old key set on that Sub-Stream.
 - The Port must respond by returning an K_GOSTOP_ACK once it is capable of receiving another IDE_KM Request.
- If the Enable bit in the IDE Extended Capability for a Stream is Clear, the receipt of K_SET_GO for both receive and transmit must cause the Port to become ready to Transmit/Receive IDE TLPs for the indicated Stream and Sub-Stream within 10 ms following the receipt of the last K_SET_GO, after which system software is permitted to Set the Enable bit for the Stream. When the Enable bit is Set:
 - The Port must be capable of processing received IDE TLPs using the Key Set armed by the received K_SET_GO Request.
 - System software must ensure that the Partner Ports initiate IDE TLPs sequenced appropriately so that a Port will not receive an IDE TLP before the Enable bit has been set.
 - The Port must respond by returning an K_GOSTOP_ACK once it is capable of receiving another IDE_KM Request.

It is strongly recommended that all Sub-Streams in both directions be fully programmed before Setting the Enable bit for the Stream.

- Is permitted for the IDE_KM Requester to transmit the K_SET_STOP command, defined in § Figure 6-61 , to indicate that a Key Set must stop being used at a Port for the indicated Stream, Sub-Stream, and direction.
 - The Port implementing the responder role must invalidate and render unreadable the indicated Key Set not more than 10 ms after the receipt of the K_SET_STOP command.
 - Upon receipt of the KEY_STOP command, for the indicated Key Set and direction, all keys must be invalidated and rendered unreadable.
 - It should be observed that this action does not directly transition the Stream to Insecure, but any subsequent attempt to use the indicated Key Set will result in the Stream transitioning to Insecure.
 - The Port must respond by returning an K_GOSTOP_ACK once it is capable of receiving another IDE_KM Request.
- When an error is detected in a received K_SET_GO or K_SET_STOP (such as an invalid $\uparrow\downarrow$ Stream ID), $\uparrow\downarrow$ it is recommended that no K_GOSTOP_ACK be returned.
- When using DOE for IDE_KM, or when IDE is enabled/disabled using the Enable bit for an IDE Stream, the following rules apply:
 - For a Configuration Request that triggers the start IDE, the Port must first return the Configuration Completion as a non-IDE TLP, and then trigger the start of IDE.
 - For a Configuration Request that stops IDE, the Port must first return the Configuration Completion as an IDE TLP, and then stop IDE.

An IDE error condition will occur if system software fails to ensure the correct sequencing of IDE enablement at the two Partner Ports.

- For a given IDE Stream, once a secure [SPDM] session has been used to respond to one QUERY or KEY_PROG request:
 - While the secure [SPDM] session that was used for initial key programming remains open, all QUERY and/or KEY_PROG requests that are received through a different secure [SPDM] session must be discarded by the Responder, and must not result in a response.
 - If the secure [SPDM] session that was used for initial key programming is closed, any subsequent QUERY and/or KEY_PROG requests received through a different secure [SPDM] session must first cause the responder to invalidate and render unreadable all keys for the IDE Stream, then transition that IDE Stream to the Insecure state, and only then respond to the QUERY/KEY_PROG request, unless it can be ensured through implementation specific means that the new session has been established with the same requester as performed the initial key programming.

| | | | |
|-----------|----------|-------------------------|-------------------------|
| 31 | 16 15 | 8 7 | 0 |
| PortIndex | Reserved | Object ID 00h: QUERY | Protocol ID 00h: IDE |

Figure 6-56 Query (QUERY) Data Object §

Base 6.4 vs Base 6.3

| 31 | 24 23 | 16 15 | 8 7 | 0 |
|--|----------|------------------------------|---|---|
| PortIndex | Reserved | Object ID 01h: QUERY_RESP | Protocol ID 00h: IDE | |
| MaxPortIndex | Segment | Bus Number | Non-ARI: Dev./Fn Num. ARI: Function Number | |
| IDE Capability Register | | | | |
| IDE Control Register | | | | |
| Link IDE Stream Control Register | | | | |
| Link IDE Stream Status Register | | | | |
| Selective IDE Stream Capability Register | | | | |
| Selective IDE Stream Control Register | | | | |
| Selective IDE Stream Status Register | | | | |
| IDE RID Association Register 1 | | | | |
| IDE RID Association Register 2 | | | | |
| IDE Address Association Register 1 | | | | |
| IDE Address Association Register 2 | | | | |
| IDE Address Association Register 3 | | | | |

Figure 6-57 Query Response (QUERY_RESP) Data Object §

| Key Set RxTxB | | 31 | 24 | 23 | 22 | 18 | 17 | 16 | 15 | 12 | 11 | 8 | 7 | 0 |
|------------------|---|----------------|---------------|----|----|----------------------------|----|----|---------------|-------------------------|----|---|---|---|
| Reserved | | | | | | Object ID 02h: KEY_PROG | | | | Protocol ID 00h: IDE | | | | |
| PortIndex | R | Key Sub-Stream | R | | | | | | | | | | | |
| Key_DW7_Byte3 | | | Key_DW7_Byte2 | | | Key_DW7_Byte1 | | | Key_DW7_Byte0 | | | | | |
| Key_DW6_Byte3 | | | Key_DW6_Byte2 | | | Key_DW6_Byte1 | | | Key_DW6_Byte0 | | | | | |
| Key_DW5_Byte3 | | | Key_DW5_Byte2 | | | Key_DW5_Byte1 | | | Key_DW5_Byte0 | | | | | |
| Key_DW4_Byte3 | | | Key_DW4_Byte2 | | | Key_DW4_Byte1 | | | Key_DW4_Byte0 | | | | | |
| Key_DW3_Byte3 | | | Key_DW3_Byte2 | | | Key_DW3_Byte1 | | | Key_DW3_Byte0 | | | | | |
| Key_DW2_Byte3 | | | Key_DW2_Byte2 | | | Key_DW2_Byte1 | | | Key_DW2_Byte0 | | | | | |
| Key_DW1_Byte3 | | | Key_DW1_Byte2 | | | Key_DW1_Byte1 | | | Key_DW1_Byte0 | | | | | |
| Key_DW0_Byte3 | | | Key_DW0_Byte2 | | | Key_DW0_Byte1 | | | Key_DW0_Byte0 | | | | | |
| IFV_DW1_Byte3 | | | IFV_DW1_Byte2 | | | IFV_DW1_Byte1 | | | IFV_DW1_Byte0 | | | | | |
| IFV_DW0_Byte3 | | | IFV_DW0_Byte2 | | | IFV_DW0_Byte1 | | | IFV_DW0_Byte0 | | | | | |

Figure 6-58 Key Programming (KEY_PROG) Data Object with Example 256b Key §

Base 6.4 vs Base 6.3

| | | | | |
|-----------|----------|--------------------------|-------------------------|------------------|
| 31 | 24 23 22 | 18 17 16 15 | 8 7 | 0 |
| Reserved | | Object ID 03h: KP_ACK | Protocol ID 00h: IDE | |
| PortIndex | R | Key Sub-Stream | R | Status Stream ID |

↑↑
Key Set
RxTxB

Figure 6-59 Key Programming Acknowledgement (KP_ACK) Data Object §

| | | | | |
|-----------|----------|----------------------------|-------------------------|--------------------|
| 31 | 24 23 22 | 17 16 15 | 8 7 | 0 |
| Reserved | | Object ID 04h: K_SET_GO | Protocol ID 00h: IDE | |
| PortIndex | R | Key Sub-Stream | R | Reserved Stream ID |

↑↑
Key Set
RxTxB

Figure 6-60 Key Set Go (K_SET_GO) Data Object §

| | | | | |
|-----------|----------|------------------------------|-------------------------|--------------------|
| 31 | 24 23 22 | 17 16 15 | 8 7 | 0 |
| Reserved | | Object ID 05h: K_SET_STOP | Protocol ID 00h: IDE | |
| PortIndex | R | Key Sub-Stream | R | Reserved Stream ID |

↑↑
Key Set
RxTxB

Figure 6-61 Key Set Stop (K_SET_STOP) Data Object §

| | | | | |
|-----------|----------|--------------------------------|-------------------------|--------------------|
| 31 | 24 23 22 | 17 16 15 | 8 7 | 0 |
| Reserved | | Object ID 06h: K_GOSTOP_ACK | Protocol ID 00h: IDE | |
| PortIndex | R | Key Sub-Stream | R | Reserved Stream ID |

↑↑
Key Set
RxTxB

Figure 6-62 Key Set Go/Stop Acknowledgement (K_GOSTOP_ACK) Data Object §

For implementations not requiring interoperable authentication and key exchange, it is permitted to use other mechanisms for key management.

Rules related to keys:

- Key size must be 256 bits.
- Following key exchange, implementation specific means must be used to ensure key security - the specific requirements for maintaining key security are platform and use case specific, and are out of scope for this document.
- Separately generated keys must be used for the Transmitter and the Receiver, and for each Sub-Stream.
 - It is strongly recommended that all keys for all Streams be separately generated.
- To support key updates without requiring an active IDE Stream to be put into a quiescent state, two key sets are defined, and the appropriate key set indicated in the IDE TLP Prefix (NFM)/ OHC-C (FM) by the Transmitter via the K bit.
 - The initial value of the K bit is permitted to be 0b or 1b, although it is recommended the initial value be 0b. Software must ensure that the selected key set has been provided with keys in both Partner Ports.
 - Once the Transmitter has indicated a change of key set via the K bit, the Receiver must mark the other key set/bank invalid until it is reprogrammed.
 - The specific requirements for the frequency of key updates are determined by platform security requirements that are outside the scope of this document.
 - It is generally recommended that hardware provide enough key storage and management resources to support changing the keys for at least one active Stream without disruption to IDE operation.
 - It is expected that key update operations for multiple streams may need to be performed by firmware/software in a serialized fashion based on hardware key storage and management resource limitations.

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: UNDERSTANDING THE IDE KEY MANAGEMENT FLOW §

The following diagram illustrates a detailed example key programming flow using the IDE_KM protocol defined above, although it should be understood that there are many possible variations on this flow.

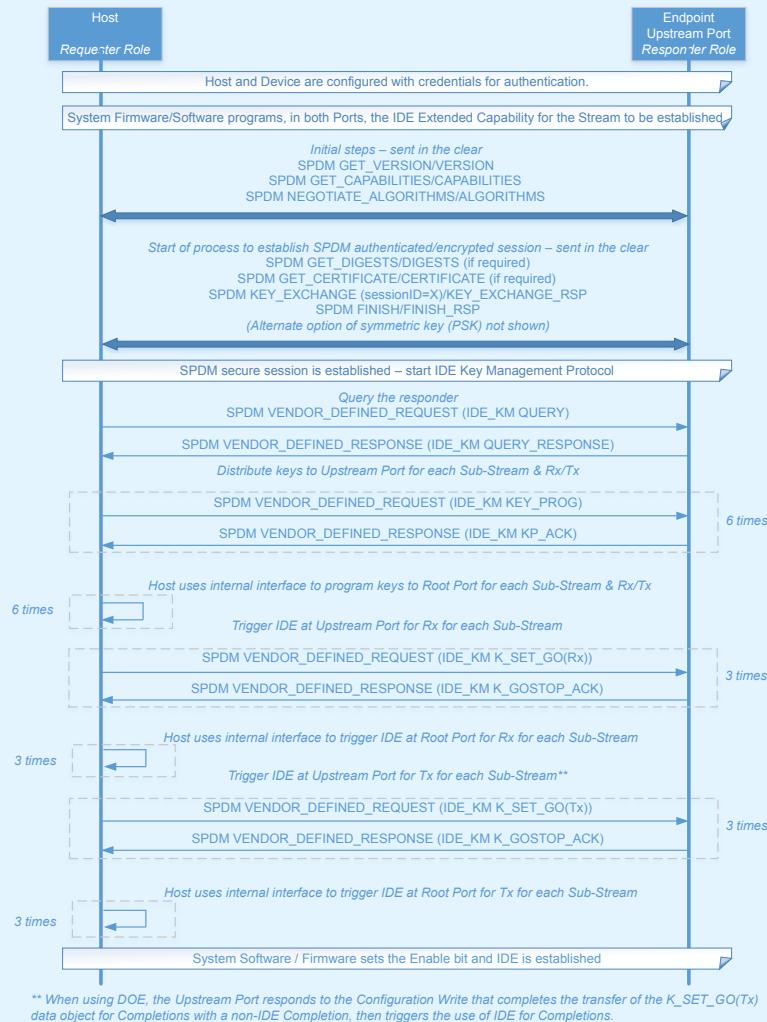


Figure 6-63 IDE_KM Example §

6.33.4 IDE TLPs §

TLPs secured by IDE are called IDE TLPs. In Non-Flit Mode, all IDE TLPs must use the IDE prefix (see [Figure 6-64](#)), and this prefix must precede all other End-End TLP prefixes. In Flit Mode, all IDE TLPs must include OHC-C.

```
↓↓↓ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 32, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": ["1", "0", "0"], "name": "Fmt", "attr": "ro" }, { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "value": ["1", "0", "0", "1"], "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 7, "value": ["1", "0", "0", "1", "0"], "name": "PR_Sent_Counter", "attr": "ro" }, { "lsbyte": 2, "lsbit": 0, "msbyte": 2, "msbit": 7, "value": ["1", "0", "0", "1", "0"], "name": "Stream_ID", "attr": "ro" }, { "lsbyte": 3, "lsbit": 3, "msbyte": 3, "msbit": 6, "value": ["1", "0", "0", "1", "0"], "name": "Sub_Stream", "addClass": "regFieldVerySmallText", "attr": "ro" }, { "lsbyte": 3, "lsbit": 3, "msbyte": 3, "msbit": 3, "value": ["1", "0", "0", "1", "0"], "name": "M", "attr": "ro" }, { "lsbyte": 3, "lsbit": 1, "msbyte": 3, "msbit": 1, "value": ["1", "0", "0", "1", "0"], "name": "K", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 0, "msbit": 0, "value": ["1", "0", "0", "1", "0"], "name": "T", "attr": "ro" } ] } ↓
```

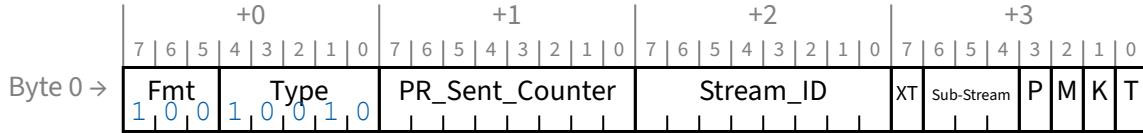


Figure 6-64 IDE TLP Prefix (NFM)

The IDE Prefix (NFM) includes:

- M bit – When Set, indicates this TLP includes a MAC
 - When aggregation is not used, the M bit must be Set for all IDE TLPs.
 - Rules for the use of aggregation are given below.
- K bit – Indicates the key set used for this TLP
 - After Transmitting a TLP with the K bit toggled for any Sub-Stream, subsequent TLP Transmissions for different Sub-Streams of the same Stream must also use the new value for the K bit.
 - After receiving a TLP with the K bit toggled, the Receiver must transition to the new key and IV set for that TLP and all subsequent TLPs associated with the Sub-Stream, and must mark the old key and IV set invalid until reprogrammed.
 - Such transitions must not affect other Sub-Streams of the IDE Stream at the Receiver.
- **XT and T bits** – **When Set, indicates The XT (eXTended TEE) bit and the T (TEE) bit indicate the TLP originated from within a trusted execution environment (TEE) attributes associated with the TLP** (see [§ Section 11.1](#)). ECN: Base 6.3 XT
 - If the TEE-IO Supported bit in the Device Capabilities Register is Clear:
 - It is permitted for IDE TLPs to originate from both trusted and non-trusted execution environments, and the value of the XT and T bit does not modify the handling of TLPs within IDE per se; the rules for trusted execution environments are not defined in this document.
 - The XT and T bit must be Clear unless the use of the XT and T bit has been explicitly defined by TEE management mechanisms outside the scope of this document.
 - If the TEE-IO Supported bit in the Device Capabilities Register is Set, **this bit** must be used for TEE management mechanisms as defined in [§ Chapter 11](#). ECN: Base 6.3 XT
- P bit – When Set, indicates the TLP includes PCRC.
 - Must only be Set when the M bit is also Set.
- Sub-Stream[2:0] – Indicates the Sub-Stream identifier value¹⁵⁵

155. In earlier versions of this specification, the Sub-Stream field was 4 bits. See [§ Section 6.33.5](#).

ECN: Base 6.3 XT

- Stream_ID[7:0] – Indicates the associated $\downarrow\downarrow$ Stream ID $\downarrow\downarrow$ $\uparrow\uparrow$ Stream_ID $\uparrow\uparrow$ value
- PR_Sent_Counter[7:0] – For non-UIO Non-Posted Requests and Completions the value must be determined according to the rules below. This field must be Reserved for Posted Requests and for UIO Requests/Completions.

In Flit Mode:

- The presence of MAC and/or PCRC are indicated using the TS field.
- The K bit, $\uparrow\uparrow$ XT bit $\uparrow\uparrow$, T bit, Sub-Stream, Stream_ID, and PR_Sent_Counter are included in OHC-C, and have the same meaning as in the IDE Prefix.

ECN: Base 6.3 XT Δ $\triangleleft\triangleright$

IDE uses Galois/Counter Mode (GCM) as defined in [AES-GCM], referred to as AES-GCM. For IDE TLPs, TLP data payload content forms the “Plaintext”, also known as P , as defined in [AES-GCM], and the TLP Header and certain other elements (defined below) form the “Additional Authenticated Data”, also known as A , as defined in [AES-GCM].

The Message Authentication Code (MAC)¹⁵⁶ size, also known as t , as defined in [AES-GCM], must be 96b (see § Figure 6-65).

```
 $\uparrow\downarrow$ { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 96, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "name": "MAC[95:88]", "attr": "ro" }, { "lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 7, "name": "MAC[87:80]", "attr": "ro" }, { "lsbyte": 2, "lsbit": 0, "msbyte": 2, "msbit": 7, "name": "MAC[79:72]", "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 3, "msbit": 7, "name": "MAC[71:64]", "attr": "ro" }, { "lsbyte": 4, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "MAC[63:56]", "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 5, "msbit": 7, "name": "MAC[55:48]", "attr": "ro" }, { "lsbyte": 6, "lsbit": 0, "msbyte": 6, "msbit": 7, "name": "MAC[47:40]", "attr": "ro" }, { "lsbyte": 7, "lsbit": 0, "msbyte": 7, "msbit": 7, "name": "MAC[39:32]", "attr": "ro" }, { "lsbyte": 8, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "MAC[31:24]", "attr": "ro" }, { "lsbyte": 9, "lsbit": 0, "msbyte": 9, "msbit": 7, "name": "MAC[23:16]", "attr": "ro" }, { "lsbyte": 10, "lsbit": 0, "msbyte": 10, "msbit": 7, "name": "MAC[15:8]", "attr": "ro" }, { "lsbyte": 11, "lsbit": 0, "msbyte": 11, "msbit": 7, "name": "MAC[7:0]", "attr": "ro" } ] } $\downarrow\uparrow$ 
```

| | +0 | +1 | +2 | +3 |
|----------|------------|------------|------------|------------|
| Byte 0 → | MAC[95:88] | MAC[87:80] | MAC[79:72] | MAC[71:64] |
| Byte 4 → | MAC[63:56] | MAC[55:48] | MAC[47:40] | MAC[39:32] |
| Byte 8 → | MAC[31:24] | MAC[23:16] | MAC[15:8] | MAC[7:0] |

Figure 6-65 MAC Layout \S

Flow Control credit accounting for IDE TLPs must handle the MAC as covered by the Header Credit.

For IDE TLPs, AES-GCM can be applied to each IDE TLP, or aggregation can be used to apply AES-GCM to multiple IDE TLPs, reducing the per-TLP overhead for the IDE TLP MAC. For a Link IDE Stream, local prefixes must be covered by the MAC (see § Figure 6-66 , § Figure 6-67), § Figure 6-70 , and § Figure 6-71). For Selective IDE Streams, local prefixes must not be covered by the MAC (see § Figure 6-68 , § Figure 6-69 , § Figure 6-72 , and § Figure 6-73).

156. Referred to as “authentication tag” or T in [AES-GCM] but renamed here to avoid confusion with other uses of “tag”

Base 6.4 vs Base 6.3

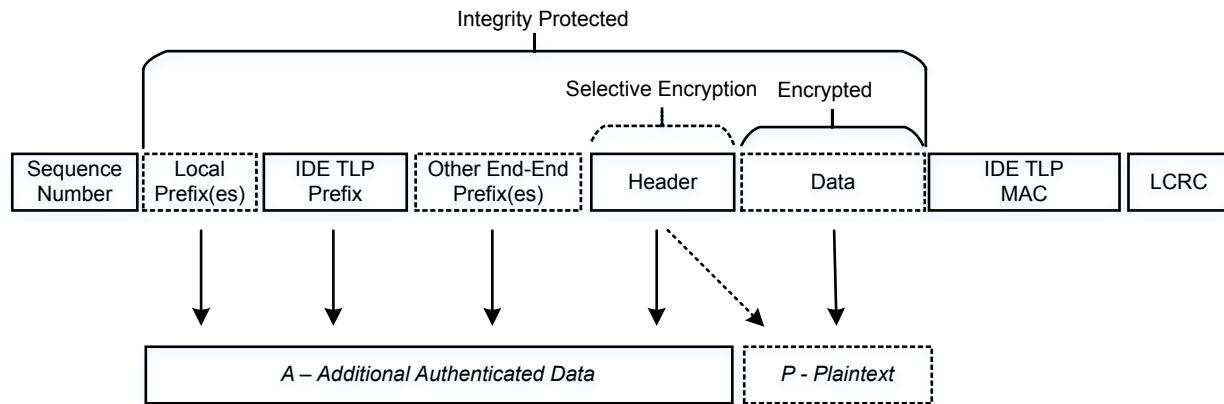


Figure 6-66 Example of IDE TLP for a Link IDE Stream without Aggregation (Non-Flit Mode) §

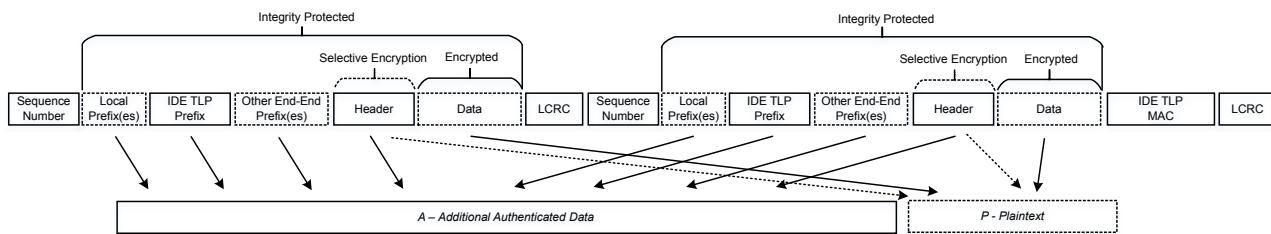


Figure 6-67 IDE TLP – Example Showing Aggregation of Two TLPs for a Link IDE Stream (Non-Flit Mode) §

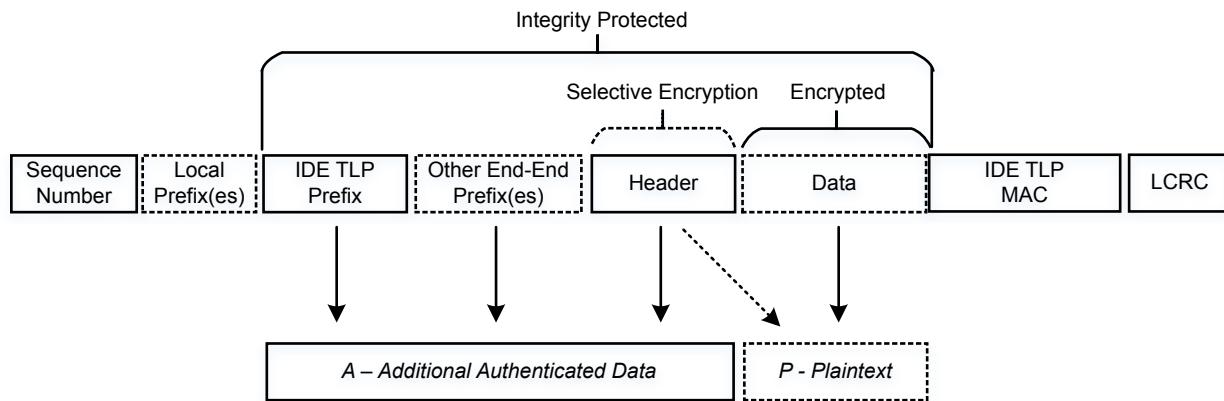


Figure 6-68 IDE TLP – Example of IDE TLP for a Selective IDE Stream without Aggregation (Non-Flit Mode) §

Base 6.4 vs Base 6.3

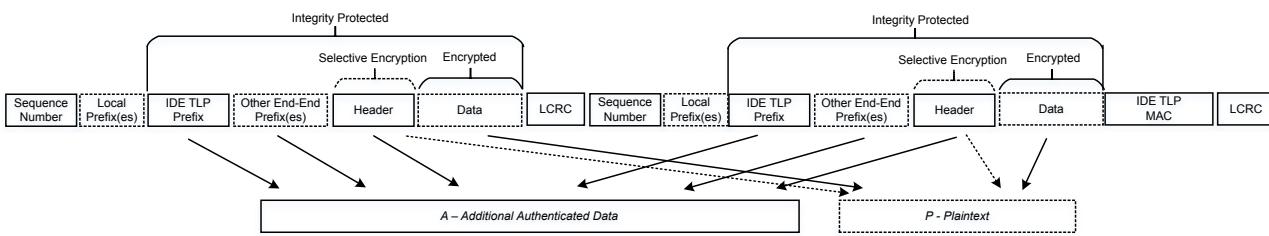


Figure 6-69 IDE TLP – Example Showing Aggregation of Two TLPs for a Selective IDE Stream (Non-Flit Mode) §

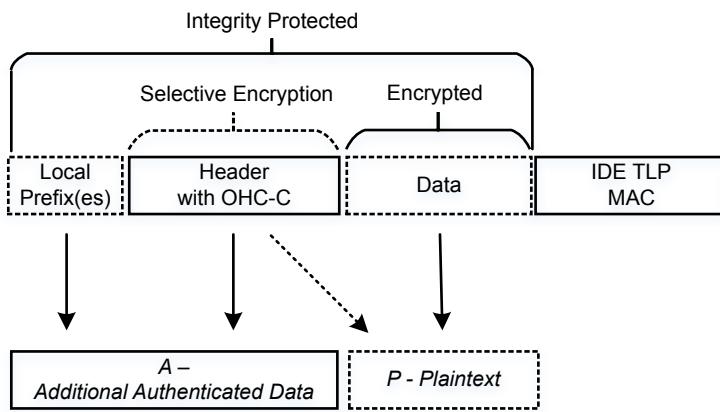


Figure 6-70 Example of IDE TLP for a Link IDE Stream without Aggregation (Flit Mode) §

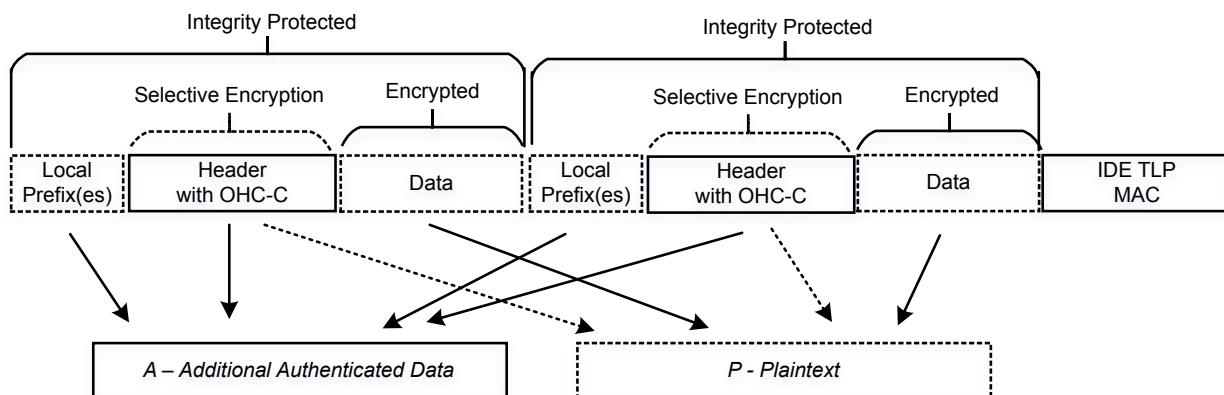


Figure 6-71 IDE TLP – Example Showing Aggregation of Two TLPs for a Link IDE Stream (Flit Mode) §

Base 6.4 vs Base 6.3

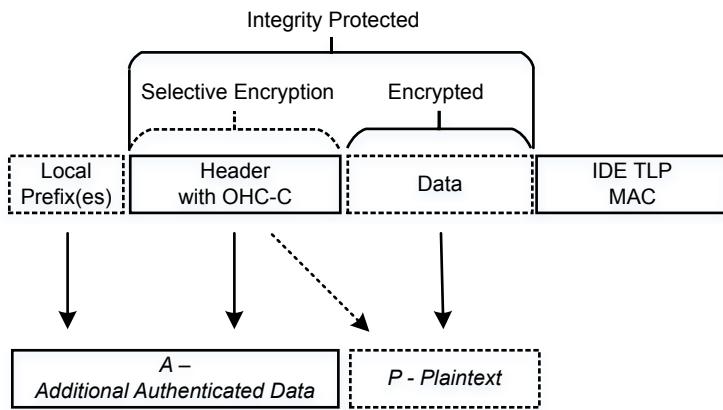


Figure 6-72 IDE TLP – Example of IDE TLP for a Selective IDE Stream without Aggregation (Flit Mode) §

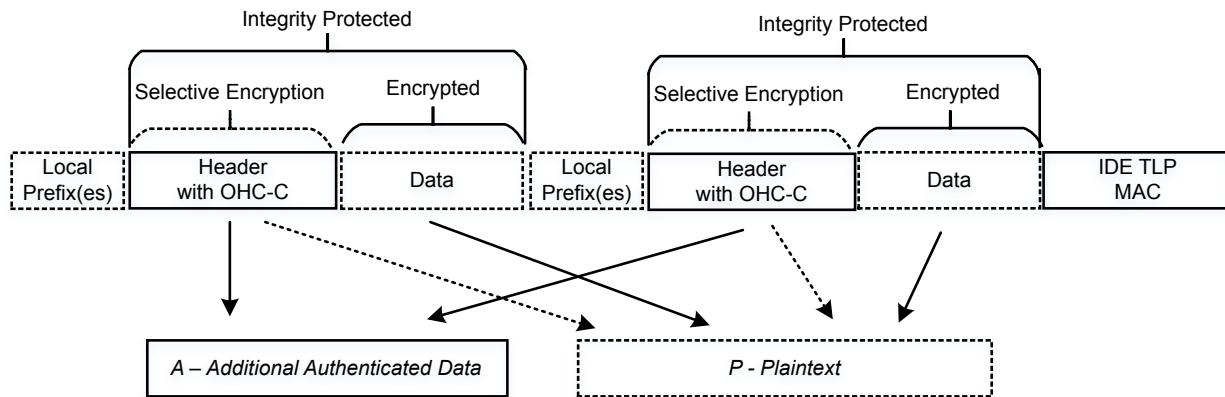


Figure 6-73 IDE TLP – Example Showing Aggregation of Two TLPs for a Selective IDE Stream (Flit Mode) §

The inputs *A* and *P* must be formed by concatenating the included TLP content in Byte order as defined in § Section 2.1.2 . Although the *A* and *P* content is conceptually concatenated as illustrated in these figures, the content placement in the IDE TLPs is the same as in non-IDE TLPs. Once the *A* and *P* content is constructed, [AES-GCM] defines how *A* and *P* must be padded – this padding is not illustrated here, and the padding is used in the [AES-GCM] calculations but is not included in the TLPs transmitted/received. When aggregation is used, the *A* and *P* content for aggregated TLPs is conceptually concatenated, for each type of content, prior to padding.

Partial header encryption provides the ability to reduce potential exposure to side-channel attacks by encrypting some portions of the Header of an IDE Memory Request while maintaining information that is required for TLP routing and low-level TLP processing in the clear. § Figure 6-74 illustrates, at a high level, the application of partial header encryption.

Base 6.4 vs Base 6.3

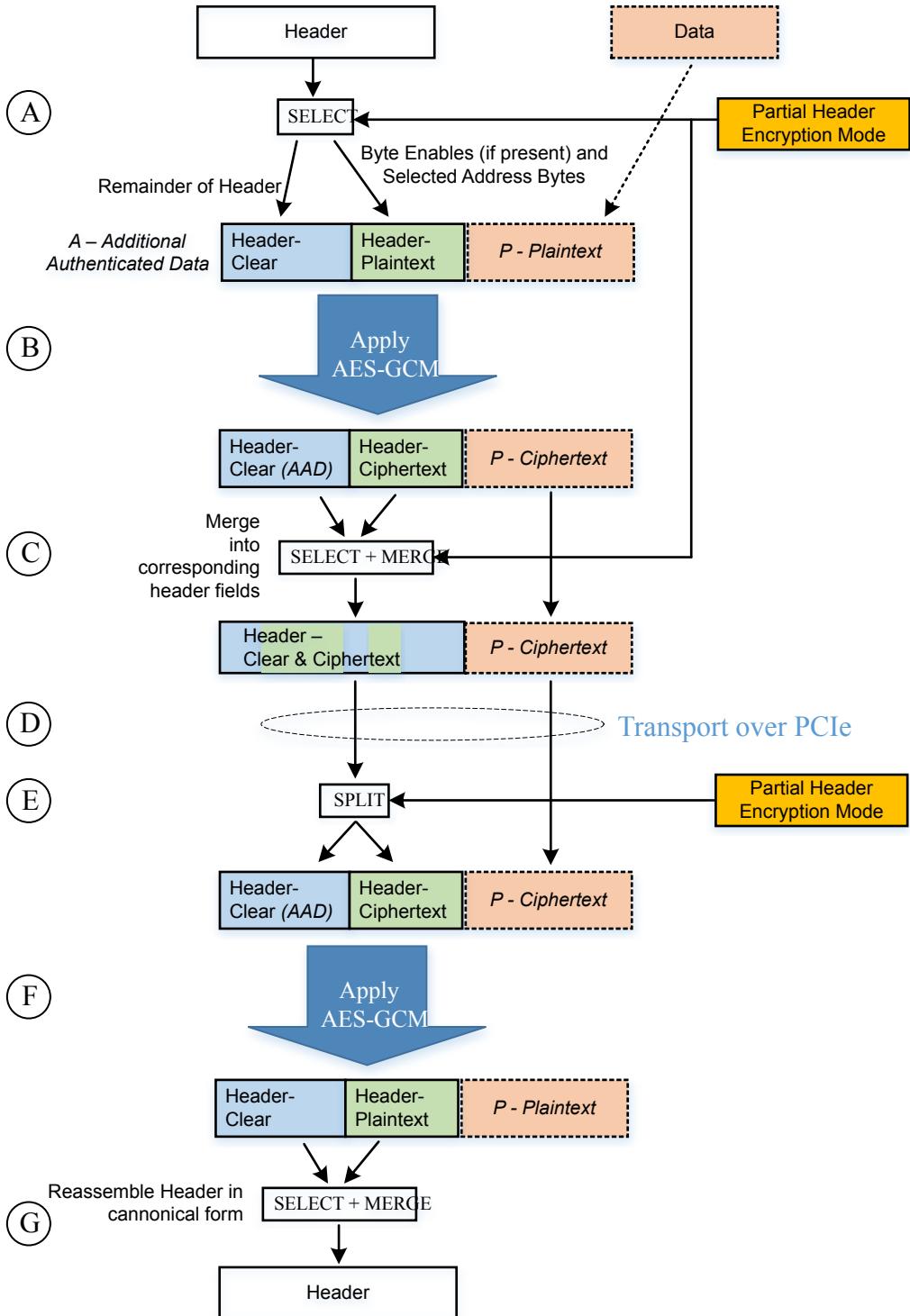


Figure 6-74 High Level Flow For Partial Header Encryption §

In partial header encryption, the encrypted portions of the Header are determined by the setting in the Partial Header Encryption Mode field of the Selective IDE Stream Control Register or Link IDE Control Register.

Rules for partial header encryption:

- When encrypted, the First DW BE and Last DW BE fields must be in the first byte of P
 - In NFM, the First DW BE and Last DW BE fields must be encrypted in all Memory Requests, except

↑↓for↓ ↑↑for:↑

 - AtomicOp

↑↓Requests,↓ ↑↑Requests↑
 - Translation

↑↓Requests, and↓ ↑↑Requests↑
 - Memory Read/DMWr Requests with the TH bit

↑↓Set,↓ ↑↑Set↑
 - In FM, for Memory Requests, if OHC-A1 is present, then the First DW BE and Last DW BE fields must be encrypted.
- Address bits selected for encryption must follow the First DW BE and Last DW BE fields, if included, in P , and are formed as:
 - Address[17:2]:
 - Byte +0: Address[17:10]
 - Byte +1: Address[9:2]
 - Address[25:2]:
 - Byte +0: Address[25:18]
 - Byte +1: Address[17:10]
 - Byte +2: Address[9:2]
 - Address[33:2]:
 - Byte +0: Address[33:26]
 - Byte +1: Address[25:18]
 - Byte +2: Address[17:10]
 - Byte +3: Address[9:2]
 - Address[41:2]:
 - Byte +0: Address[41:34]
 - Byte +1: Address[33:26]
 - Byte +2: Address[25:18]
 - Byte +3: Address[17:10]
 - Byte +4: Address[9:2]
- ↑↑In the case of Translation Requests, header Byte 14 and header Byte 15 (see § Section 10.2.2) are handled in the same way as for Memory Reads, including the Reserved bits and the CXL Src bit.↑

 - At the Transmitter, the Header content selected for encryption is concatenated at the front of P and removed from A , increasing the size of P and decreasing the size of A accordingly.
 - If PCRC is enabled when using partial header encryption:
 - Relative ordering of bits for PCRC input is maintained, with the same Header content used as for P .
 - For the PCRC calculation only, the selected portions of the header must be padded to 64bits with 0's in the most significant bits.

 Errata: Base 6.3
 B811△↔

 Errata: Base 6.3
 B811△↔

- At the Receiver, the operation is reversed in order to apply AES-GCM to the *A* and *C* content and then finally the complete header is reconstructed.
- The PCRC is calculated at the Receiver using the decrypted *P* content and the Header portion padded to 64bits with 0's in the most significant bits.
- When an error causes an IDE TLP with partial header encryption to be logged in the Header Log Register, the header fields selected for partial header encryption are permitted to contain all 0's or the encrypted values, but must not contain the decrypted values unless through ~~Implementation-specific~~. ~~Implementation-specific~~ means it is assured that doing so will not violate any security requirements.

§ Figure 6-75 through § Figure 6-78 illustrate the application of partial header encryption.

Base 6.4 vs Base 6.3

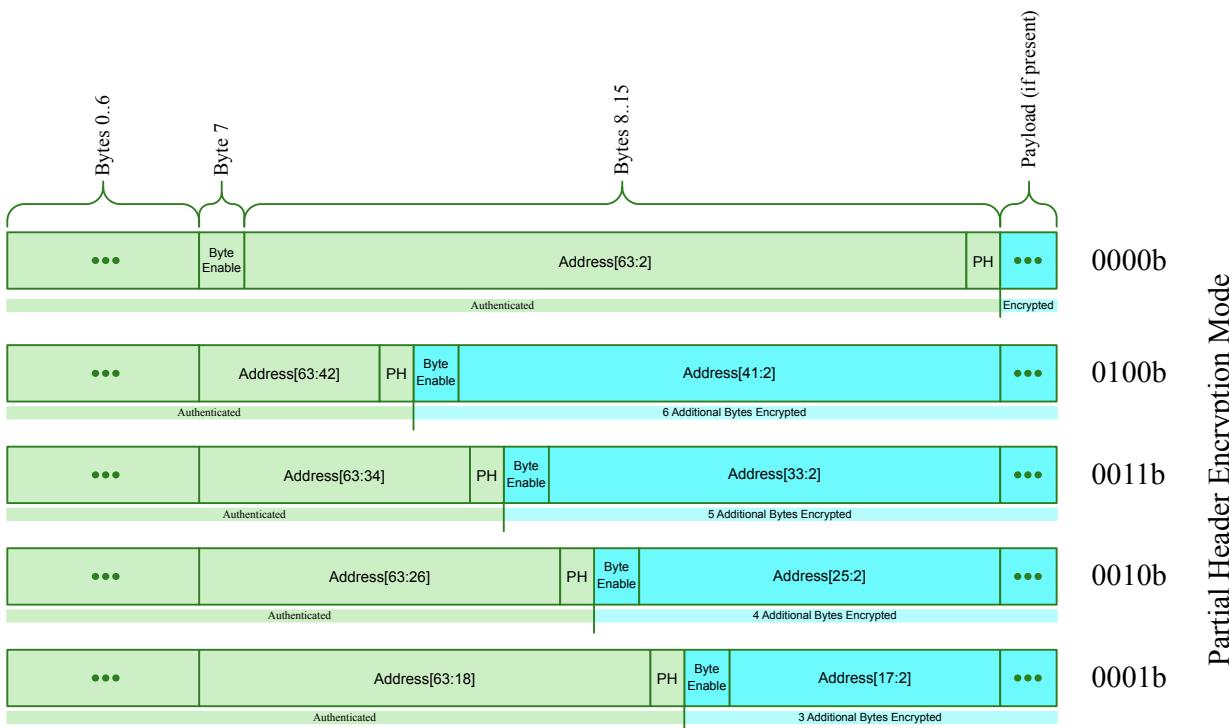


Figure 6-75 Partial Header Encryption in NFM with Byte Enables §

Base 6.4 vs Base 6.3

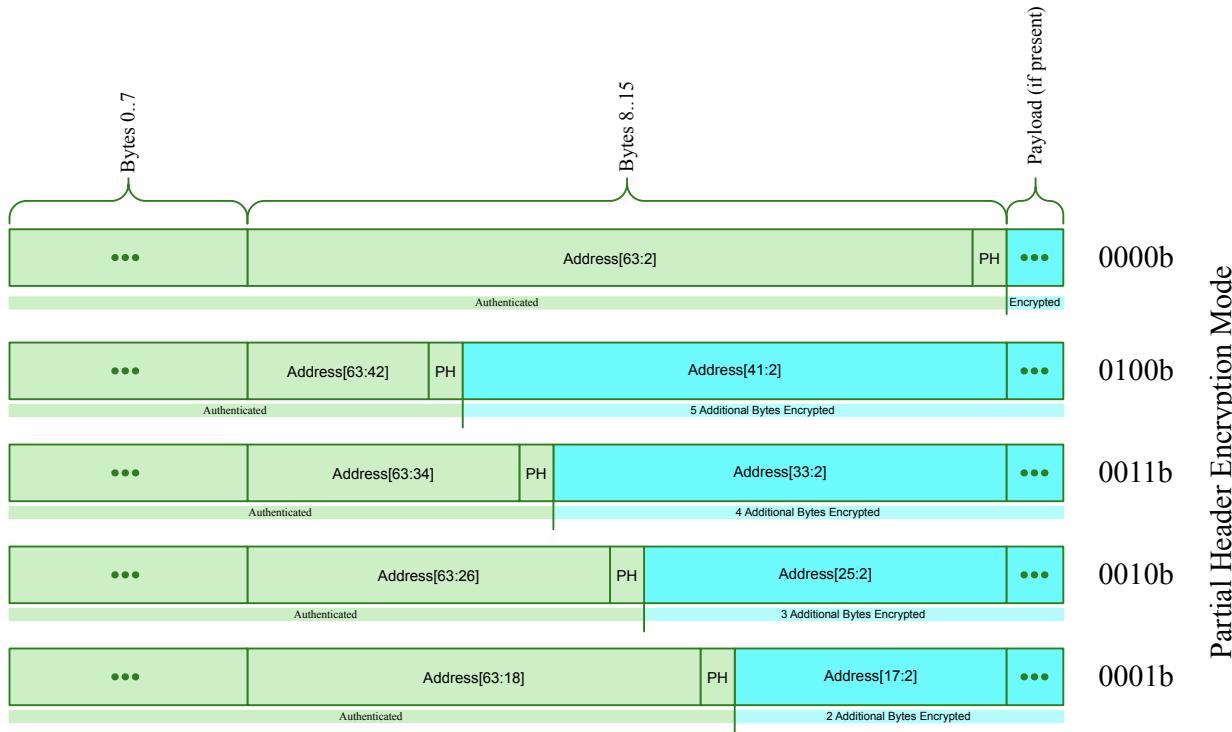


Figure 6-76 Partial Header Encryption in NFM without Byte Enables §

Base 6.4 vs Base 6.3

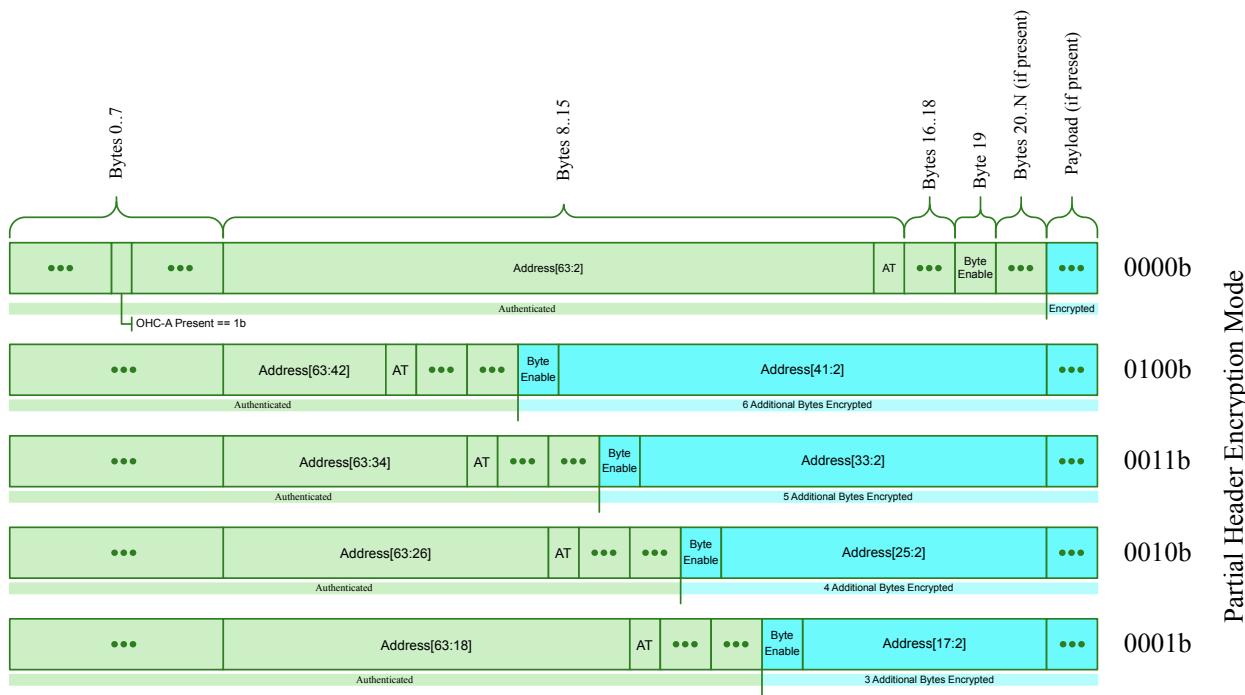


Figure 6-77 Partial Header Encryption in FM with OHC-A1 §

Partial Header Encryption Mode

Base 6.4 vs Base 6.3

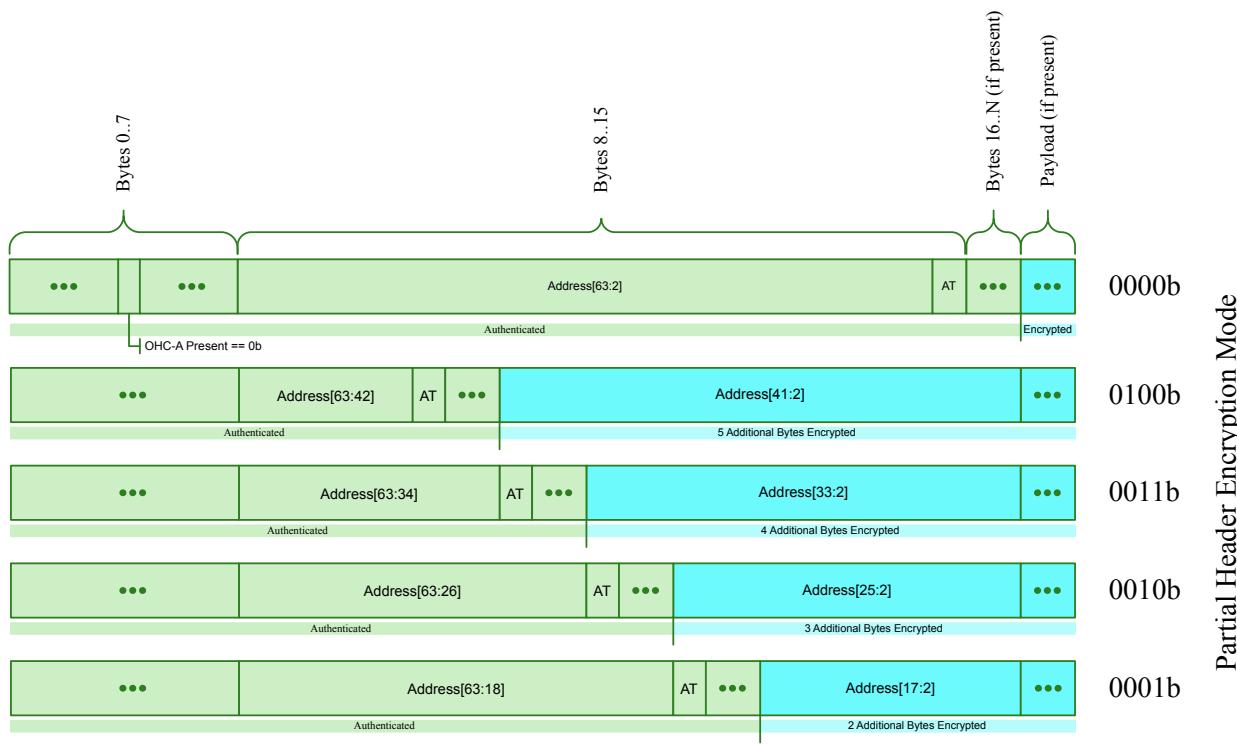


Figure 6-78 Partial Header Encryption in FM without OHC-A1 §

The use of Local TLP Prefixes with IDE TLPs is permitted. Local TLP Prefixes, if present, must precede the IDE Prefix (NFM). For Link IDE Streams, Local TLP Prefixes must be included in A. For Selective IDE Streams, Local TLP Prefixes must not be included in A or P.

The IDE Prefix (NFM) must be included in A. All OHC content (FM) must be included in A.

In NFM, the use of other End-End TLP Prefixes, besides the IDE Prefix, with IDE TLPs is permitted. End-End TLP Prefixes, if present, must follow the IDE Prefix, and must be included in A.

When aggregation is used, all TLPs associated with a single MAC are considered to be part of an “aggregated unit.”

As defined in [AES-GCM], a single invocation is performed for each TLP when aggregation is not used, and for each aggregated unit when aggregation is used.

As with all TLPs, IDE TLPs are covered by Data Link Layer mechanisms, such that physical Link errors are detected and corrected before received TLPs are presented to the Receiver’s cryptographic processing mechanisms.

The use of ECRC with IDE TLPs is not permitted. If ECRC is enabled for non-IDE TLPs, then IDE TLPs must be formed as if ECRC were not enabled, and the TD bit in the TLP Header must be Clear.

To enable the detection of faults in the encryption/decryption logic, which occur outside of the path protected by the MAC, IDE implementations are permitted optionally to support the Plaintext CRC (PCRC) mechanism, for which the following rules apply:

- Software must only enable PCRC when both Partner Ports support the PCRC mechanism.
 - It is permitted to enable the PCRC mechanism on a per-IDE Stream basis.

- When PCRC is enabled for an IDE Stream, all Transmitted TLPs associated with that Stream that include a MAC must also include PCRC, if and only if there is *P* content in the TLP or aggregated unit of TLPs.
 - In NFM, presence of a MAC is indicated by the M bit in the IDE Prefix being Set and presence of the PCRC is indicated by the P bit in the IDE Prefix being Set
 - In FM, the TS field is used to indicate presence of MAC / PCRC (see § [Section 2.2.1.2](#)).
 - When aggregation is used, the TLPs of an aggregated unit that do not include a MAC also must not include PCRC (see § [Figure 6-79](#)).

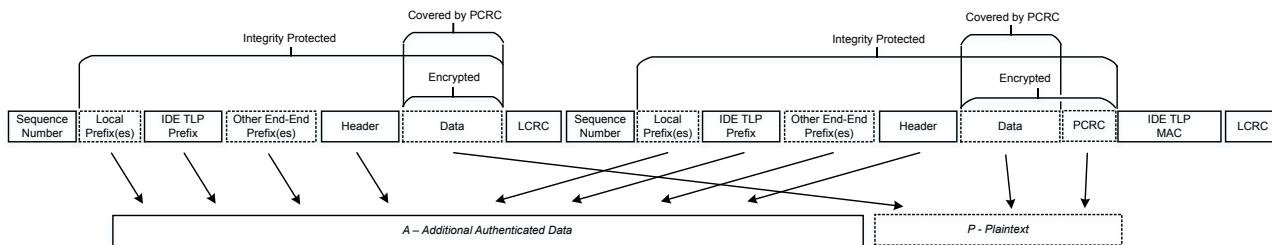


Figure 6-79 Example Illustrating PCRC Application to Two Aggregated IDE TLPs for a Link IDE Stream (NFM) §

- Base 6.4 vs Base 6.3
- In NFM, when PCRC is enabled for an IDE Stream, the ultimate Receiver must check that all received TLPs or aggregated units associated with that Stream that include *P* content and a MAC (as indicated by the M bit in the IDE Prefix) also have the P bit in the IDE Prefix Set.
 - If, for a TLP, the P bit in the IDE Prefix is Clear and the M bit is Set, the Receiver must report this as a PCRC Check Failed error.
 - PCRC must be calculated across all *P* content¹⁵⁷ for a given invocation of AES-GCM, and per the following:
 - The polynomial used has coefficients expressed as 04C1 1DB7h
 - The seed value must be FFFF FFFFh
 - All *P* content must be included in the PCRC calculation
 - PCRC calculation starts with bit 0 of byte 0 and proceeds from bit 0 to bit 7 of each byte of *P*
 - The result of the PCRC calculation must be complemented, mapped as shown in § [Table 2-55](#) (following the same mapping as for ECRC), and appended to the *P* content, and encrypted/decrypted along with the other *P* content, with the encrypted PCRC value appended to the other *P* content and preceding the MAC (see § [Figure 6-79](#)).
 - The PCRC must only be checked by the ultimate Receiver of the IDE TLP including PCRC.
 - A failure of the PCRC check indicates that one or more bits of the data payload have been corrupted — the Receiver's use of the data payload is outside the scope of this specification, but it is strongly recommended that the corrupted data not be used as if it were uncorrupted.
 - PCRC Check Failed is a reported error.
 - Flow Control credit accounting for IDE TLPs that include PCRC must handle the PCRC as covered by the Header Credit.

¹⁵⁷. As with ECRC, the PCRC is calculated using the TLP as-transmitted, including, for any data payload, all bytes as-transmitted regardless of the byte enable values.

IMPLEMENTATION NOTE: PCRC REUSE OF ECRC LOGIC §

In most respects, PCRC is calculated in the same way as ECRC, and the CRC polynomial is the same.

In some implementations it may be possible to ~~↑↓re-use↓↑↑reuse↑~~ the same logic for PCRC calculation as is used for ECRC, when both PCRC and ECRC are supported.

ECRC is not allowed on IDE TLPs, and so no TLP could have both PCRC and ECRC.

These rules apply to Selective IDE Stream TLPs:

- § Table 6-35 defines which TLP types are permitted for a Selective IDE Stream.
 - TLP types that are not permitted for a Selective IDE Stream are still permitted to be used, and can be secured using Link IDE, if enabled.
- Receipt of an IDE TLP associated with a Selective IDE Stream that is not a permitted TLP Type is an IDE Check Failed error.
 - The Receiver must transition to Insecure for the associated IDE Stream.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).

Table 6-35 TLP Types for Selective IDE Streams §

| TLP Type | Description | Permitted for Selective IDE Streams |
|----------|------------------------------------|--|
| MRd | Memory Read Request | Y |
| MRdLk | Memory Read Request-Locked | Y |
| MWr | Memory Write Request | Y |
| IORD | I/O Read Request | N |
| IOWr | I/O Write Request | N |
| CfgRd0 | Type 0 Configuration Read Request | N |
| CfgWr0 | Type 0 Configuration Write Request | N |
| CfgRd1 | Type 1 Configuration Read Request | Y |
| CfgWr1 | Type 1 Configuration Write Request | Y |
| DMWr | Deferrable Memory Write | Y |
| Msg | Message Request | Y when Routed by ID, Routed by Address, or Routed to Root complex ¹⁵⁸ N for other routing mechanisms |
| MsgD | Message Request with data | Y when Routed by ID, Routed by Address, or Routed to Root complex ¹⁵⁹ N for other routing mechanisms |

158. Routed to Root Complex is permitted only when the Partner Port is a Root Port, as indicated by the Default Stream bit being Set.

159. Routed to Root Complex is permitted only when the Partner Port is a Root Port, as indicated by the Default Stream bit being Set.

| TLP Type | Description | Permitted for Selective IDE Streams |
|--------------------------------|--|---|
| Cpl | Completion without Data | Y |
| CplD | Completion with Data. | Y |
| CplLk | Completion for Locked Memory Read without Data | Y |
| CplDLk | Completion for Locked Memory Read – otherwise like CplD. | Y |
| FetchAdd | Fetch and Add AtomicOp Request | Y |
| Swap | Unconditional Swap AtomicOp Request | Y |
| CAS | Compare and Swap AtomicOp Request | Y |
| LPrfx | Local TLP Prefix | Allowed to be present, but not secured if TLP is associated with a Selective IDE Stream |
| EPrfx (applicable in NFM only) | End-End TLP Prefix | Y if the TLP is part of a Selective IDE Stream |

- All IDE TLPs must be associated with an IDE Stream, identified via an IDE $\downarrow\downarrow$ Stream ID $\downarrow\downarrow$ $\uparrow\uparrow$ Stream_ID $\uparrow\uparrow$
 - Software must assign IDE Stream IDs such that two Partner Ports use the same value for a given IDE Stream.
 - Software must assign IDE Stream IDs such that every enabled IDE Stream associated with a given terminal Port is assigned a unique $\downarrow\downarrow$ Stream ID $\downarrow\downarrow$ $\uparrow\uparrow$ Stream_ID $\uparrow\uparrow$ value at that Port
 - It is permitted for a platform to further restrict the assignment of Stream IDs.
- When only a Link IDE Stream is enabled for a particular TC, all TLPs using that TC must be secured using the corresponding Link IDE Stream.
- For a Transmitter to associate a specific TLP with a specific Selective IDE Stream, the following criteria must be satisfied:
 - The Selective IDE Stream Enable bit in the Selective IDE Stream Control Register must be Set.
 - The TLP type must be permitted for Selective IDE Streams (see § Table 6-35).
 - The TC of the TLP must match the TC value in the Selective IDE Stream Control Register .
 - For a Configuration Request, the Selective IDE for Configuration Requests Enable bit must be Set (applicable to Root Ports only).
 - For a Completion, the $\downarrow\downarrow$ Stream ID $\downarrow\downarrow$ $\uparrow\uparrow$ Stream_ID $\uparrow\uparrow$ must match those in the corresponding $\downarrow\downarrow$ Non Posted $\downarrow\downarrow$ Request.
 - If an ACS mechanism in the Port redirects the TLP towards the Root Complex, the Default Stream bit must be Set, indicating that the Selective IDE Stream targets the Root Complex. See § Section 6.12.3 .
 - For a Routed-to-Root-Complex Message, the Default Stream bit must be Set, indicating that the Selective IDE Stream targets the Root Complex.
 - For an ID-Routed $\downarrow\downarrow$ Message, the $\downarrow\downarrow$ $\uparrow\uparrow$ Message not associated with a Default Stream, unless there is an exception based on implementation-specific criteria: $\uparrow\uparrow$
 - $\uparrow\uparrow$ The $\uparrow\uparrow$ destination RID must be greater than or equal to the RID Base and less than or equal to the RID Limit in the Selective IDE RID Association Register block, $\downarrow\downarrow$ unless: The

Errata: Base 6.3
B834 Δ \triangleleft Errata: Base 6.3
B811 Δ \triangleleft

Base 6.4 vs Base 6.3

ID-Routed Message is associated with a Default Stream, in which case the destination RID must be ignored, or ↑↑and↑

- ↑↑there is an exception made based on implementation-specific criteria. Additionally, in ↑↑In↑ Flit Mode, the destination Segment value must also match the value of Segment Base in the Selective IDE RID Association Register block.

- Unless there is an exception made via implementation specific means, when ATS is supported and enabled, all Translation Requests and Untranslated Memory Requests are associated with the default Stream, otherwise for a Memory Request not already associated with a Default Stream, the destination address must be greater than or equal to the Memory Base value and less than or equal to Memory Limit value in a Selective IDE Address Association Register block (as applies when targeting a specific Function's BAR or the Base/Limit range of addresses assigned to a Device)¹⁶⁰.
- For a TLP not already associated with any other Stream, the Default Stream bit must be Set.
- The TLP is selected or precluded through implementation specific means. For example, the Transmitter could associate all Memory Requests initiated by one or more internal Functions with a specific Selective IDE Stream, particularly when it is known that the Partner Port is a Root Port, and that all Requests initiated by that internal Function target system memory.
- If the ↑↑XT Enable bit in the Selective IDE Stream Control Register is Clear, and the TEE-Limited Stream bit in the Selective IDE Stream Control Register is Set, ECN: Base 6.3 XT△↔ Requests with the T bit Set are associated with this Selective IDE Stream.
 - Requests that otherwise would be associated with this Selective IDE Stream and have the T bit Clear must be Transmitted as non-IDE TLPs if Link IDE is not enabled, or as Link IDE TLPs if Link IDE is enabled.
- ↑↑If both the XT Enable bit and the TEE-Limited Stream bit are Set in the Selective IDE Stream Control Register , Requests with the XT bit and/or the T bit Set are associated with this Selective IDE Stream.↑ ECN: Base 6.3 XT△↔
 - ↑↑Requests that otherwise would be associated with this Selective IDE Stream and have the XT bit Clear and the T bit Clear must be Transmitted as non- IDE TLPs if Link IDE is not enabled, or as Link IDE TLPs if Link IDE is enabled.↑
- TLPs not determined by the Transmitter to be associated with a specific Selective IDE Stream or that are precluded through ↑↑implementation specific ↑↑implementation specific means, must not be associated with any Selective IDE Stream.
- If supported, the TEE-Limited Stream bit must be configured in both Partner Ports before enabling a Selective IDE Stream.
 - It is permitted for the Partner Ports to have different values for the TEE-Limited Stream bit.
- Separate TCs must use separate Selective IDE Streams.
- Software must ensure that Selective IDE Streams are assigned to RID and address ranges that are not overlapping.
 - If software violates this rule by programming overlapping ranges, hardware must associate a given TLP with one Selective IDE Stream matching the RID/address range, but it is implementation specific which Selective IDE Stream from the overlapping set is associated.

These rules apply to IDE Fail Messages:

- Receivers must accept IDE Fail Messages received as IDE TLPs and also as non-IDE TLPs.

¹⁶⁰. As with the Base/Limit address routing mechanism for Type 1 (Bridge) Functions, the IDE Address Association mechanism does not consider the value of the AT field to determine TLP routing.

- When an IDE Fail Message is to be transmitted on an enabled Stream that is in the Insecure state, the Message must be transmitted as a non-IDE TLP. This includes cases where the transition of that Stream to Insecure was the trigger for the IDE Fail Message.
- When an IDE Fail Message is to be transmitted on an enabled Stream that is in the Secure state, the Message must be transmitted as an IDE TLP. This includes cases where the transition to Insecure on a Selective IDE Stream is the trigger for the IDE Fail Message, and the transmission Stream is either a different Selective IDE Stream or a Link IDE Stream.

These rules apply to TLPs other than IDE Fail Messages:

- When ready to schedule a TLP to be Transmitted that is associated with an enabled Stream that is in the Insecure state, the Transmitter must instead discard the TLP.
~~↑↓When Link IDE is Enabled and an IDE TLP has already been received for a given TC, and a TLP is received on that TC as a non-IDE TLP, the Receiver must discard the received TLP.~~
- ~~↑↓↑~~ Errata: Base 6.3
B822 $\triangleleft\triangleright$
- When both Link IDE Stream and one or more Selective IDE ~~↑↓Stream~~ Stream(s) are enabled at the same Port, for Transmitted TLPs the Selective IDE Stream(s) take precedence: selected TLPs must be associated with the Selective IDE Stream(s); TLPs not associated with any Selective IDE Stream must use Link IDE.
 - If both Link IDE and Selective IDE are enabled, and a TLP to be Transmitted is associated with an enabled Selective IDE Stream that is in the Insecure state, the Transmitter must not use Link IDE for the TLP and must instead discard the TLP.
- At the ultimate Receiver, IDE TLPs must be associated with the ~~↑↓Stream ID~~ ~~↑↓Stream_ID~~ indicated in the IDE Prefix (NFM)/ OHC-C (FM).
- Once Link IDE has been enabled (see § Section 6.33.3 ~~↑↓~~, ~~↑↓~~ for a particular TC, ↑ for each Sub-Stream of the Stream, until the Port receives an IDE TLP on that ~~↑↓TC and~~ ~~that~~ Sub-Stream, the Port must continue to accept non-IDE TLPs ~~↑↓on that TC~~ of the FC type corresponding to that Sub-Stream.
 - Once the Port receives an IDE TLP on a Sub-Stream it must ~~↑↓silently~~ discard ~~↑↓subsequent~~ non-IDE TLPs on that Sub-Stream for as long as the associated Stream is enabled. ~~↑↓This silent discard operation takes precedence over reporting errors associated with these non-IDE TLPs (e.g., ECRC, Poison)~~.¹⁶¹
- For Root Ports, when Selective IDE for Configuration Requests Enable for a particular Selective IDE Stream is Set, all Configuration Requests that match that Selective IDE Stream must be Transmitted as IDE TLPs associated with that Selective IDE Stream.
 - After enabling Selective IDE for Configuration Requests it is recommended that system software perform a Configuration Request using that Selective IDE Stream, so that the Partner Port will be triggered to reject subsequent Configuration Requests not associated with the Selective IDE Stream.
 - How the Root Complex ensures that only authorized system software generates Configuration Requests is outside the scope of this document.
- Once Selective IDE for Configuration Requests Enable for a particular Selective IDE Stream is Set by system software, it is strongly recommended that system software not Clear the bit for as long as the Selective IDE Stream itself is enabled.
- Specific Selective IDE Streams on Root Ports for which Selective IDE for Configuration Requests Enable is Set must transmit Configuration Requests that are associated with those Selective IDE Streams as Type 1 Configuration Requests.

¹⁶¹ ~~↑↓Earlier versions of this specification required discarding non-IDE TLPs but did not specify whether error reports were suppressed. Such behavior is still permitted but it is recommended to lower the impact of non-IDE TLPs by avoiding consequences (e.g., reporting ECRC errors).~~

- Configuration Requests associated with a Selective IDE Stream will be received at an Upstream Port as Type 1 Configuration Requests.
- Configuration Requests received as IDE TLPs that are directed to pass through the Switch must pass without modification through a Switch when Flow-Through IDE Stream Enabled is Set in both the Upstream Port and the Egress Downstream Port.
- Configuration Requests that are not directed to pass through a Switch must be accepted by the Receiver provided the Target RID is an implemented Function either associated with that Upstream Port or a Downstream Port of a Switch in that Upstream Port. If the Target RID is not an implemented Function, the Configuration Request must be handled as an Unsupported Request.
- For Upstream Ports that are the IDE Terminus of a Selective IDE Stream with Selective IDE for Configuration Requests Supported Set, until the Port receives a Configuration Request as an IDE TLP on that Selective IDE Stream, the Port must continue to accept Configuration Requests as non-IDE TLPs. Once a Configuration Request has been received as an IDE TLP on that Selective IDE Stream, then the Port must accept only Configuration Requests associated with that Selective IDE Stream, and must discard all Configuration Requests received on other IDE Streams or as non-IDE TLPs, for as long as that Selective IDE Stream is enabled.
- For Selective IDE, for TLP types other than Configuration Requests, the requirements for rejecting TLPs are determined by the TEE associated with the Selective IDE Stream (see § [Section 6.33.1](#)).
- The use of TLP poisoning, as indicated by the EP bit in the TLP header being Set, is permitted for IDE TLPs when applied by the originating Transmitter.
- For IDE TLPs, it is not permitted to modify any part of a TLP, including the EP bit, at any intermediate point between the two Partner Ports.
- Software must configure Selective IDE such that all Links on the entire path between the Partner Ports are operating either in FM, or in NFM.
- If TEE-IO Supported is Set, for an Endpoint Upstream Port:
- In Flit Mode, if Segment Captured is Set¹⁶², on a Selective IDE Stream:
 - A Requester must include OHC-C with the Requester Segment Valid (RSV) bit Set.
 - A Completer must include OHC-A5.
 - A Completer receiving a Request on a Selective IDE Stream other than the Default Stream must handle it as an Unsupported Request if the Requester ID field of the Request is less than the RID Base or greater than the RID Limit in the Selective IDE RID Association Register block of the Stream.
 - In Flit Mode, the Requester Segment value (if included in the Request) must also match the value of Segment Base in the Selective IDE RID Association Register block.
 - A Requester receiving a Completion on a Selective IDE Stream other than the Default Stream must handle it as an Unexpected Completion if the Completer ID field of the Completion is less than the RID Base or greater than the RID Limit in the Selective IDE RID Association Register block of the Stream.
 - In Flit Mode, the Completer Segment value (if included in the Completion) must also match the value of Segment Base in the Selective IDE RID Association Register block.
- If TEE-IO Supported is Set, for a Root Port:
- A Completer receiving a Request on a Selective IDE Stream for which the Root Port is an IDE Terminus must handle it as an Unsupported Request if the Requester ID field of the Request is less than the RID Base or greater than the RID Limit in the Selective IDE RID Association Register block of the Stream.
 - In Flit Mode, if the Requester Segment Valid (RSV) bit is Set, the Requester Segment value must also match the value of Segment Base in the Selective IDE RID Association Register block.

¹⁶². The value of Segment Captured indicates that Segment values are being applied – the captured Segment value is not used for the tests defined here.

- A Requester receiving a Completion on a Selective IDE Stream for which the Root Port is an IDE Terminus must handle it as an Unexpected Completion if the Completer ID field of the Completion is less than the RID Base or greater than the RID Limit in the Selective IDE RID Association Register block of the Stream.
 - In Flit Mode, if the Completion includes OHC-A5, the Completer Segment value must also match the value of Segment Base in the Selective IDE RID Association Register block.
- **If the XT Supported bit in the IDE Capability Register is Set:**
 - **If the XT Enable bit for the associated IDE Stream is Clear:**
 - **The Transmitter must Clear the XT bit in the Transmitted IDE TLPs associated with the Stream.**
 - **The Receiver should process the TLP as if the XT bit was Clear.**
 - **If the XT Enable bit for the associated IDE Stream is Set:**
 - **The Transmitter must include the XT bit in the Transmitted IDE TLPs as defined by the TEE management mechanisms in § Chapter 11..**
 - **The Receiver must process the XT bit in the Received IDE TLPs as defined by the TEE management mechanisms in § Chapter 11..**
- **If the XT Supported bit in the IDE Capability Register is Clear:**
 - **The Transmitter must Clear the XT bit in the Transmitted IDE TLPs associated with the Stream.**
 - **The Receiver should process the TLP as if the XT bit was Clear.**

ECN: Base 6.3 XT $\triangleleft\triangleright$ ECN: Base 6.3 XT $\triangleleft\triangleright$

The use of Multicast (see § Section 6.14) with Selective IDE Streams is outside the scope of this specification.

6.33.5 IDE TLP Sub-Streams §

With [AES-GCM] it is desirable to maintain TLPs in-order so that the Transmitter and Receiver can independently maintain IV in sync with each other without the overhead required to transmit the IV for each TLP or aggregated unit. However, certain TLP bypassing is required for deadlock avoidance, and this is reflected in the different types of Flow Control Credit Types – Posted Request header/data payload, Non-Posted Request header/data payload, and Completion header/data payload (see § Section 2.6.1). To provide in-order TLP processing where possible, and to simplify implementations that structure their internal buffering according to these Flow Control Credit types, IDE introduces the concept of a Sub-Stream within which TLP traffic is maintained fully in-order between the IDE Partner Ports. It is ensured within IDE Sub-Streams that TLPs travel in-order between the IDE Partner Ports both to reduce the per-TLP overhead as noted, and also to ensure that certain attack scenarios, such as the reordering or replaying by an Adversary-in-the-Middle, of Posted Requests, will be detected by the Receiver without additional TLP tracking logic.

With [AES-GCM] it is essential to maintain synchronization between the Partner Ports such that all TLPs associated with an IDE Stream are always routed from one Partner Port to the other. If any IDE TLPs are misrouted the result will typically be an unrecoverable error for the associated IDE Stream (see detailed requirements below).

Each IDE Stream includes Sub-Streams distinguished by TLP **credit** type and direction, for which the following rules apply:

Errata: Base 6.3
B834 $\triangleleft\triangleright$

- For each Sub-Stream there is a Sub-Stream identifier:
 - 000b – Posted Requests **and UIO Writes**
 - 001b – Non-Posted Requests **and UIO Reads**
 - 010b - Completions
 - Values 011b-110b are Reserved

Errata: Base 6.3
B834 $\triangleleft\triangleright$

- 111b - In NFM, Reserved; In FM, indicates a TLP that includes OHC-C but is not an IDE TLP.
 - In earlier versions of this specification, ~~↑↑the↑ Sub-Stream ↑↑field↑ was 4 bits in Symbol 3, bits 7:4. ↓↓Bit 7 is now Reserved.↓~~ If the TEE-IO Supported bit in the Device Capabilities Register is Set, components must ~~↑↑implement↑~~
~~↑↑not treat↑~~ bit 7 as ~~↑↑Reserved.↓~~ ~~↑↑part of the Sub-Stream field.↑~~ If ~~↑↑the↑~~ TEE-IO Supported ~~↑↑bit in the Device Capabilities Register↑~~ is Clear, components are permitted ~~↑↑— though strongly discouraged —~~ to treat bit 7 as part of ~~↑↑the↑~~ Sub-Stream ~~↑↑field.↑~~
- Errata: Base 6.3
B834Δ
- For each Sub-Stream, per [AES-GCM], there must be a 96b initialization vector IV of deterministic construction, consisting of:
 - a fixed field in bits 95:64 of the IV, where bits 95:64 are all 0's
 - an invocation field in bits 63:0 of the IV, containing the value of a counter, initially set to the value 0000_0001h for each Sub-Stream upon establishment of the Stream and each time the Sub-Stream key is refreshed, and incremented every time an IV is consumed.
 - Each Sub-Stream must support the use of its own unique key value and invocation field initial counter value.
- § Section 6.33.7 defines additional requirements for Switches and Root Complexes that support peer-to-peer routing of Selective IDE Streams.
- The ordering rules defined in § Section 2.4 define constraints on TLP/TLP ordering, but do not provide mechanisms to detect improper reordering. With IDE, counters are used to enable the ultimate Receiver to detect improper reordering of non-UIO TLPs while allowing permitted reordering. These counters and associated mechanisms, including error checks, do not apply to UIO TLPs. Separate sets of these counters must be operated for each IDE Stream, and operated according to the following rules:
- For the Transmitter, two 8 bit counters must be maintained: a count of Posted Requests Transmitted since the last Non-Posted Request or IDE Sync Message was Transmitted, called PR_Sent_Counter-NPR, and a count of Posted Requests Transmitted since the last Completion or IDE Sync Message was Transmitted, called PR_Sent_Counter-CPL
 - Upon entry to the Secure state, both counters must be initialized to 0.
 - Both counters must be incremented for each Posted Request IDE TLP Transmitted associated with the IDE Stream.
 - When a Non-Posted Request associated with the IDE Stream is Transmitted, the PR_Sent_Counter-NPR value must be used in the PR_Sent_Counter field of the IDE Prefix (NFM)/ OHC-C (FM) for the Non-Posted Request, and then PR_Sent_Counter-NPR must be reset to zero.
 - When a Completion associated with the IDE Stream is Transmitted, the PR_Sent_Counter-CPL value must be used in the PR_Sent_Counter field of the IDE Prefix (NFM)/ OHC-C (FM) for the Completion, and then PR_Sent_Counter-CPL must be reset to zero.
 - When either the PR_Sent_Counter-NPR or the PR_Sent_Counter-CPL reaches 245, an IDE Sync Message (see § Section 2.2.8.11) must be transmitted to the Partner Port as an IDE TLP.
 - It is permitted to Transmit an IDE Sync Message at other times.
 - Every time an IDE Sync Message is Transmitted to the Partner Port, both the PR_Sent_Counter-NPR and PR_Sent_Counter-CPL must be incremented prior to forming the IDE Sync Message, and then both the PR_Sent_Counter-NPR and PR_Sent_Counter-CPL must be reset to zero.
 - For the Receiver, two 64 bit counters must be maintained: a count of Posted Requests received since the last Non-Posted Request was received, called PR_Received_Counter-NPR and a count of Posted Requests received since the last Completion was received, called PR_Received_Counter-CPL
 - Upon entry to the Secure state, both counters must be initialized to zero.

- Both counters must be incremented for each Posted Request IDE TLP received associated with the IDE Stream.
- When a Non-Posted Request associated with the IDE Stream is received then the PR_Sent_Counter value carried in the IDE prefix (NFM)/ OHC-C (FM) must be subtracted from the PR_Received_Counter-NPR, and the PR_Received_Counter-NPR updated with the result. If this subtraction underflows, this is an error – see rules related to error handling below.
- When a Completion associated with the IDE Stream is received then the PR_Sent_Counter value carried in the IDE prefix (NFM)/ OHC-C (FM) must be subtracted from the PR_Received_Counter-CPL, and the PR_Received_Counter-CPL updated with the result. If this subtraction underflows, this is an error – see rules related to error handling below.
- When an IDE Sync Message associated with the IDE Stream is received then:
 - the PR_Sent_Counter-NPR value carried in the IDE Stream Sync Message must be subtracted from the PR_Received_Counter-NPR, and the PR_Received_Counter-NPR updated with the result,
 - the PR_Sent_Counter-CPL value carried in the IDE Stream Sync Message must be subtracted from the PR_Received_Counter-CPL, and the PR_Received_Counter-CPL updated with the result.

If either subtraction underflows, this is an error – see rules related to error handling in § Section 6.33.7.

IMPLEMENTATION NOTE: DETECTION OF IMPROPER REORDERING §

As with other TLPs, IDE TLPs need to be reordered to satisfy requirements for deadlock avoidance, but some other forms of reordering are forbidden as IDE TLPs pass over PCIe between Ports. These ordering requirements are defined in § Table 6-36, and are stated in terms of Posted Requests (PR), Non-Posted Requests (NPR), and Completions (Cpl). The following examples illustrate selected reordering cases.

An attack based on TLP reordering (or a delay that has the effect of reordering) can be implemented using a variety of mechanisms that all result in the same observed behavior, and will be detected using the mechanisms defined by IDE.

§ Figure 6-80 illustrates the case where a Posted Request is allowed to bypass a Non-Posted Request, as is required for deadlock avoidance. IDE supports having Posted Requests bypass Non-Posted Requests through the use of Sub-Streams within a given Stream. There is a similar requirement for Posted Request to be able to bypass Completions (not illustrated).

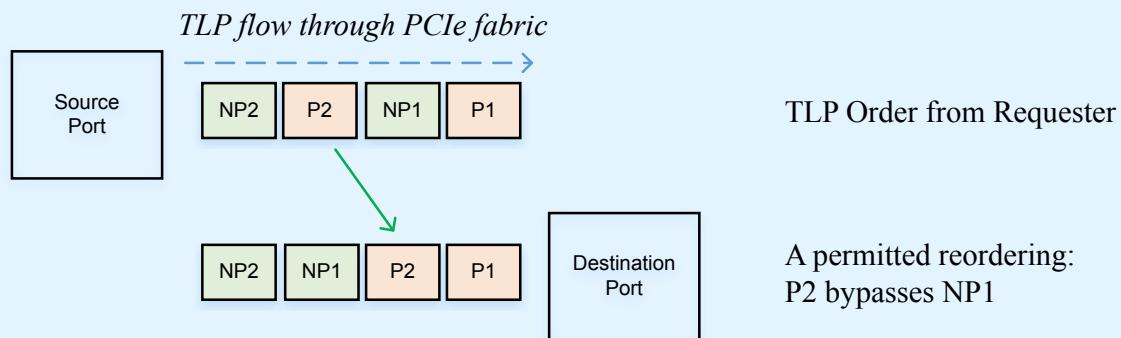


Figure 6-80 Example – Posted Requests Allowed to Bypass Non-Posted Requests §

§ Figure 6-81 illustrates a case where an attacker attempts to bypass a Posted Request with a Non-Posted Request, which could, for example, cause the consumption of stale data. This case will be detected through the use of the PR Sent Counter mechanism, through which the Transmitter at source Port associated with the Stream indicates to the Receiver at the Destination Port how many Posted Requests were transmitted between successive Non-Posted Requests. This indication is carried in the IDE TLP Prefix (NFM)/OHC-C (FM), which is integrity protected and so cannot be modified without detection at the Receiver. In this example, NP1 will carry the indication that a Posted Request (P1) was transmitted, and this will not match the Receiver's count of Posted Requests received, enabling this illegal reordering to be detected.

Base 6.4 vs Base 6.3

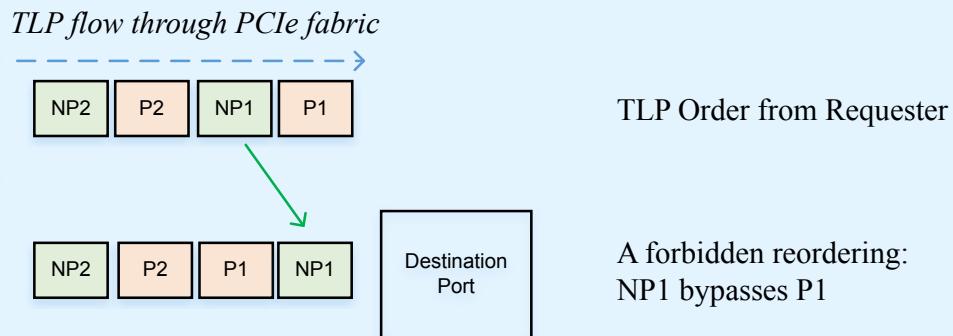


Figure 6-81 Example – Non-Posted Requests Never Allowed to Bypass Posted Requests §

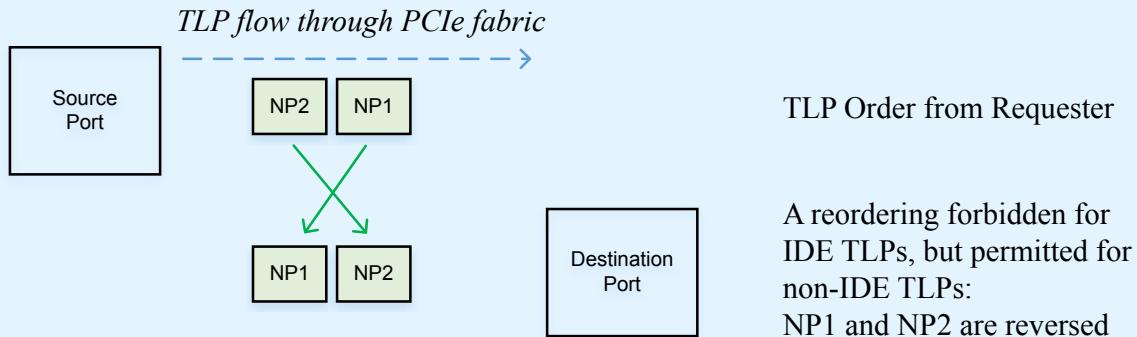


Figure 6-82 Example – Secure Non-Posted Request Reordering Not Allowed Over PCIe Fabric §

Without IDE, Non-Posted Requests are allowed to bypass each other, but within an IDE Stream, reordering of TLPs of the same type is disallowed in order to simplify the operation of the integrity/encryption mechanisms. § Figure 6-82 illustrates that this will cause the reordering of Non-Posted Requests to be detected as an integrity check failure, even though there isn't a security exposure per se.

Note that reordering attacks are possible through Retimers, Switches, and any other device or equipment that can alter the flow of TLPs at any point between the originating Port and the Destination Port.

The counters used to implement the reordering checks are of different sizes in the Transmitter and Receiver. The Transmitter counters only need to be 8 bits wide in order to accommodate the arbitrarily chosen limit of 245 that triggers the required transmission of an IDE Sync Message. Because routing elements can modify the relative order of TLPs in different Sub-Streams, Receivers cannot assume any particular limit on the amount of reordering that can occur, and so the Receiver counters were chosen to be 64 bits wide so as to ensure that under normal operating conditions it is not possible for them to overflow.

6.33.6 IDE TLP Aggregation §

The following rules relate to aggregation (see § Figure 6-67 and § Figure 6-69):

- When aggregation is supported, as indicated at the Port level by the Aggregation Supported bit in the IDE Capability Register, then a Receiver must be capable of supporting:
 - the aggregation within any Sub-Stream of up to the lesser of 8 TLPs, or as many TLPs as can be received based on the outstanding Flow Control credits applicable to that Sub-Stream
 - the receipt of other TLPs that are not part of the Sub-Stream between the TLPs of the aggregated unit.
- When aggregation is supported, as indicated at the Port level by the Aggregation Supported bit in the IDE Capability Register, then for each Sub-Stream where aggregation is to be permitted software must enable aggregation at the Transmitting Port.
- If aggregation is to be used, Software must enable aggregation prior to enabling the IDE Stream.
- When aggregation is enabled, a Transmitter:
 - must apply aggregation only among TLPs within the same Stream and Sub-Stream,
 - must aggregate at most the selected number of TLPs in the corresponding Aggregation Mode field,
 - must limit the number of TLPs aggregated such that the sum of the data payloads of all TLPs of an aggregated unit does not exceed 256 DW,
 - is permitted to Transmit other TLPs that are not part of the Sub-Stream between the TLPs of the aggregated unit,
 - must, when transmitting an IDE Sync Message, treat the IDE Sync Message as the last TLP of an aggregated unit,
 - is permitted to aggregate fewer TLPs than the number permitted.
- When aggregating TLPs, the Transmitter must include a MAC only for the last TLP of the aggregated unit, for which the included MAC must cover the A and P content for all the TLPs of the unit.
 - The Transmitter must treat a TLP as the last TLP of an aggregated unit unless the Transmitter can guarantee that it will transmit another TLP within the aggregated unit within 1 µs.
- The same key and IV set must be used for all TLPs in an aggregated unit.
- If the K bit is to be toggled, it must only be toggled for the first TLP of an aggregated unit.
 - Receivers must check for violations of this rule – a violation is an IDE Check Failed error.
 - The Receiver must transition to Insecure for the associated IDE Stream.
 - This is a reported error associated with the Receiving Port (see § Section 6.2).
- It is permitted for the TLPs of an aggregated unit to be interleaved with other TLPs, including IDE TLPs associated with other Streams, or non-IDE TLPs.
- All aggregated TLPs of a unit must be processed before it is possible for the Receiver to complete authenticated decryption of the unit.
- If an IDE TLP without a MAC is received at a Receiver where Aggregation is not supported, or if the Receiver detects a violation of any of the rules in this section, this is an IDE Check Failed error.
 - The Receiver must transition to Insecure for the associated IDE Stream.

IMPLEMENTATION NOTE: USE OF AGGREGATION §

To reduce the bandwidth overheads associated with the MAC, the use of aggregation is encouraged. Although aggregation may increase the latency for a Receiver to make use of the received TLPs, typically this increase will not be significant, and in many cases the overall performance should be improved through the use of aggregation.

Although the specific optimizations will depend on traffic types and use model requirements, typically aggregating about 4 TLPs will provide a good balance between the possible bandwidth improvement and the added latency. Typically a Transmitter policy should assume that the Receiver will buffer all the TLPs of an aggregated unit until the Receiver checks are complete before releasing any of the TLPs for further processing. Transmitters can reduce the effective latency impacts of aggregation when there is knowledge of the underlying traffic, for example by ensuring that a doorbell or other “trigger” TLP is either not aggregated, or that an aggregated unit is ended with the transmission of such TLPs.

Receivers should be designed to ensure that aggregated units of TLPs can be buffered with minimal stalling. Typically this implies that the amount of aggregation a Receivers supports should be significantly less than the amount of Receiver Flow Controller buffering advertised.

The IDE TLPs of an aggregated unit may be interleaved with other TLPs, and in some cases this may be required. For example, if we consider a case where a Switch receives a number of aggregated Memory Reads followed by a Memory Write, all of which target the same egress Port. If, at the egress Port, there are insufficient Flow Control credits to transmit all of the Memory Reads, then the Switch may be required to allow the Memory Write to bypass the blocked Memory Reads. Receivers are required to be tolerant to such “interruptions” of an aggregated unit.

6.33.7 Flow-Through Selective IDE Streams §

A Switch or Root Complex is permitted to support Flow-Through Selective IDE Streams without supporting operation as an IDE Terminus (i.e., it is permitted for a Switch or RC to have Flow-Through IDE Stream Supported Set, Link IDE Stream Supported Clear, and Selective IDE Streams Supported Clear). It is also permitted for a Switch/Root Port that is enabled for flow-through IDE to be an IDE Terminus, and in such cases, flow-through IDE TLPs must be routed without modification, and only TLPs for which the Switch/Root Port is the source or ultimate destination must be handled by the Port as an IDE Terminus.

A Switch that supports Flow-Through Selective IDE Streams must implement the IDE Extended Capability at every Port of the Switch.

Switches and RCs that support Flow-Through IDE Streams must, when enabled, implement modified ordering rules defined in § Table 6-36 , and § Table 6-37 applied per-Stream, for IDE TLPs that pass through the Switch/RC from a given Ingress Port to a given Egress Port. The entries A2, B3, B4, C3, C4, and D5 are all No to ensure that there is no reordering within any Sub-Stream. In all other cases, the rules defined in § Section 2.4 must be followed; for example: between different Stream IDs, different Ingress Ports, different Egress Ports, or between IDE TLPs and non-IDE TLPs.

Hardware is not required to follow this modified ordering model beyond the TLP flow between the two Partner Ports for an IDE Stream, e.g. within an Endpoint or RC, and so at the system level it must not be assumed that the ordering behavior observed will match the modified IDE Ordering model.

Although Switches/RCs must not reorder IDE TLPs within a Flow-Through IDE Stream based on Relaxed Ordering or IDO, including when ACS mechanisms are being used, it is permitted for those TLPs to have the RO and/or IDO bits Set .

The IDE Sync Message, like all other Messages, is a Posted Request, and the IDE protocol depends on the IDE Sync Message being ordered with all other Posted Requests for a given IDE Stream.

Table 6-36 IDE Revised Ordering Rules for Flow-Through non-UIO IDE Streams - Per Stream §

| Row Pass Column? | Posted Request (Col 2) | Non-Posted Request | | Completion (Col 5) |
|-----------------------|---------------------------|-------------------------|--------------------------|-------------------------|
| | | Read Request (Col 3) | NPR with Data (Col 4) | |
| Non-Posted Request | Posted Request (Row A) | No | Yes | Yes a) Y/N b) Yes |
| | Read Request (Row B) | No | No | No Y/N |
| | NPR with Data (Row C) | No | No | No Y/N |
| Completion (Row D) | | No | Yes | Yes No |

Table 6-37 IDE Revised Ordering Rules for Flow-Through UIO IDE Streams - Per Stream §

| Row Pass Column? | UIO PR-FC TLP (Col U1) | UIO NPR-FC TLP (Col U2) | UIO Completion (Col U3) |
|----------------------------|------------------------|-------------------------|-------------------------|
| UIO PR-FC TLP (Row UA) | No | Yes/No | Yes/No |
| UIO NPR-FC TLP (Row UB) | Yes/No | No | Yes/No |
| UIO Completion (Row UC) | Yes | Yes | No |

Switches/RCs must not modify Flow-Through IDE TLPs between the ingress and egress Ports.

Switches/RCs must only route IDE TLPs through Ports with the Flow-Through IDE Stream Enabled bit Set. If an IDE TLP is received by an Ingress Port or routed to an Egress Port, and the Port's Flow-Through IDE Stream Enabled bit is Clear, the Port must handle the TLP as a Misrouted IDE TLP error unless there is a higher precedence error. Further:

- For an Ingress Port, the Port must not forward the IDE TLP.
- For an Egress Port, the Port must not Transmit the IDE TLP.
- If the IDE TLP is a Non-Posted **↑↑Request or a UIO↑** Request, the Port must not return a Completion.

Errata: Base 6.3
B834△◀▷

The use of ACS in combination with Flow-Through IDE Streams is permitted, but requires care to ensure that the modified ordering defined above will be satisfied. It should also be understood that, because Relaxed Ordering does not apply within the modified ordering rules defined above, certain use cases such as those involving ACS P2P Completion Redirect are likely to have reduced performance.

Because hardware used with IDE will typically be required to satisfy platform-level trust requirements, in many cases ACS is not required to achieve and maintain secure platform behaviors when IDE is in use.

6.33.8 Other IDE Rules §

Rule for Non-Posted IDE [↑↑Requests and UIO↑ Requests:](#)

Errata: Base 6.3
B834△◀▷

- If the TEE-IO Supported bit in the Device Capabilities Register is Clear, Requesters of Non-Posted Requests [↑↑and UIO Requests↑](#) must check that the corresponding received Completion(s) be returned using the same [↑↓Stream ID↓](#) [↑↑Stream ID↑](#) and the same T bit value as was associated with the Non-Posted Request – a violation is an IDE Check Failed error.
 - The Requester must transition to Insecure for the associated IDE Stream.
- If the TEE-IO Supported bit in the Device Capabilities Register is Set, Requesters of Non-Posted Requests [↑↑and UIO Requests↑](#) must check that the corresponding received Completion(s) be returned using the same [↑↓Stream ID↓](#) [↑↑Stream ID↑](#) as was associated with the Non-Posted Request – a violation is an IDE Check Failed error.
 - The Requester must transition to Insecure for the associated IDE Stream

Errata: Base 6.3
B834△◀▷

Errata: Base 6.3
B834△◀▷

The following rules relate to resets:

- Any Conventional Reset to an Upstream Port or to the Bridge Function of a Downstream Port, or any FLR to a Function containing an IDE Extended Capability, must result in all IDE Streams associated with that Function transitioning to the Insecure state, and all keys must be invalidated and rendered unreadable.
 - Additional implementation specific mechanism are required in many cases to ensure security of all associated data is maintained.
- An FLR to a Function that does not contain an IDE Extended Capability must not affect IDE operation
 - In some cases IDE_KM can be affected by an FLR to a Function that does not contain an IDE Extended Capability, but does implement the IDE_KM responder role via DOE – see § Section 6.33.3.

Use of mechanisms that result in the blocking or termination of TLPs, such as exist for AtomicOps, DMWr, and the End-End TLP Prefix Blocking mechanism, must be carefully coordinated with the use of Selective IDE Streams, to avoid the dropping of Selective IDE TLPs in such a way that would result in a IDE Check Failed error.

The following rules relate to the use of Access Control Services (ACS - see § Section 6.12) with IDE.

- When Link IDE is used, ACS mechanisms are permitted to be used as architected without restrictions.
- If Selective IDE is used with ACS to enable Direct I/O as documented in the Implementation Note: ACS Redirect and Guest Physical Addresses (GPAs) (see § Section 6.12.4), ACS mechanisms are permitted to be used as architected without restrictions.
- If Selective IDE is used for P2P communication without any ACS redirect mechanisms enabled, the remaining ACS services are permitted to be used as architected without restrictions.
- Use of Selective IDE for P2P communication with ACS redirect mechanisms enabled has a number of major issues with ordering and portions of Sub-Streams taking different paths. Such use is out of scope of this specification.
- Use of Selective IDE under any use case where ACS services (or any other mechanism) blocks or otherwise terminates IDE TLPs will result in the associated Selective IDE Stream going to Insecure due to the failure of the intended ultimate destination Port to receive the blocked/terminated IDE TLP.

The following rules relate to error handling

- Receipt of a Link IDE TLP or Selective IDE TLP for which there is not an associated IDE Stream is a Misrouted IDE TLP error; this is a reported error associated with the Receiving Port.
- Receipt of a Link IDE TLP by a Switch that targets an Egress Port for which there is not a Link IDE Stream associated with the same TC and in the Secure state is a Misrouted IDE TLP error; this is a reported error associated with the Ingress Port.
- The Transaction Layer must return flow control credit and handle as a Misrouted IDE TLP, but take no other action in response to a received Misrouted IDE TLP.
- The detection of any of the following conditions is IDE Check Failed error, a reported error associated with the Receiving Port (see § Section 6.2).
 - a MAC check failure occurs when the Receiver's check of the MAC of a received TLP, or aggregated unit of TLPs, fails,
 - underflow of the PR-Received-Counter-NPR or PR_Received_Counter-CPL (indicating an improper reordering has been detected),
 - either or both of the PR-Received-Counter-NPR/PR_Received_Counter-CPL 64-bit counters overflow (indicating a failure to receive the Transmitted NPR/CPL TLPs).
 - The Sub-Stream identifier field contains a Reserved/unsupported value.

Upon detection of one or more of these conditions, the IDE Stream State Machine for this IDE Stream must enter Insecure. This is a reported error associated with the Receiving Port (see § Section 6.2). The TLP that triggered the error and all subsequent IDE TLPs received associated with the same IDE Stream must be discarded, following update of Flow Control credits, for as long as the associated Stream is enabled. There must be no additional error(s) logged for subsequent TLPs received associated with an IDE Stream already in the Insecure state. These rules also apply to TLPs received while in the Insecure state if that state was reached for reasons other than those listed above.

- Receiving an IDE TLP that is a Completion with UR or UC status is not a security error and must not by itself trigger a transition to Insecure.
- Upon detection of an error associated with an aggregated unit of TLPs, when Advanced Error Reporting is supported, only the last TLP of the unit must be logged in the Header Log Register and TLP Prefix Log Register.
- All IDE Streams associated with a Link must transition to Insecure when DL_Active transitions from asserted to deasserted for that Link.
- Upon transition from Secure to Insecure for any reason, other than that the corresponding **$\uparrow\downarrow$ Link/Selective IDE Stream Enable $\downarrow\uparrow$** bit is Cleared or the associated Traffic Class designator maps to a non-UIO VC, for a given Stream, the Port must transmit an IDE Fail Message indicating the **$\uparrow\downarrow$ Stream ID $\downarrow\uparrow$ Stream ID \uparrow** to the Partner Port
- Upon receipt of an IDE Fail Message, for the indicated Stream the Port must transition to Insecure
 - When an IDE Stream enters Insecure due to the receipt of a IDE Fail Message then that transition into Insecure must not result in the Transmission of an IDE fail message.
- Upon entry into Insecure, all active key sets and IVs for the associated IDE Stream must be marked as invalid.
- For an IDE Stream to exit Insecure and return to Secure, the IDE Stream must be re-established using a new key and IV set.
- In the Insecure state, private data associated with the affected IDE Stream(s) must be secured.
 - How this is done is implementation specific.
- To prepare to exit the Insecure state for a Stream and return to the Secure state, software must write a 0b to the corresponding **$\uparrow\downarrow$ Selective IDE Stream Enable $\downarrow\uparrow$ Selective IDE Stream Enable \uparrow** or **$\uparrow\downarrow$ Link IDE Stream Enable $\downarrow\uparrow$ Link IDE Stream Enable \uparrow** bit, even if it is already 0b.

- Hardware must not return a Stream to the Secure state until the Enable bit has been written to 0b and subsequently Set.
- Additional actions not defined here are typically necessary based on specific use model requirements.

To return a Link IDE Stream for TC0/VC0 to the Secure state, it is necessary to reset the Device, e.g. using a Hot Reset, to make it possible for Configuration Requests/Completions to pass across the Link, unless the Device provides an alternate, implementation-specific, mechanism.

- When processing received IDE TLPs, all error checks must be completed, or an equivalent delay must be inserted, prior to signaling an error, such that it is not possible for an external observer to determine at which stage in error checking the error(s) was(were) detected.

The following rules relate to Power Management:

- The No_Soft_Reset bit must be Set
- All state related to keys and counters must be maintained in D0 , D1 , D2 and D3_{hot} .
 - The IDE Extended Capability is subject to the same rules as other register structures defined by this specification, and because it is itself essential for IDE operation, any condition where the IDE Extended Capability programming is lost also necessarily results in a loss of the ability to maintain IDE operation; Such conditions must result in all IDE Streams transitioning to the Insecure state, and all keys must be invalidated and rendered unreadable.
- It is permitted to support retention of state related to keys and counters in D3_{cold} , however such mechanisms are beyond the scope of this document.

The following rules relate to maintaining a secure local environment:

- Attempts to modify IDE registers, BARs, and other structures that could affect the security of the device or an IDE Stream must be detected
 - The resulting actions are implementation specific
- It is permitted to enter Insecure when a condition is detected that could affect the security of the device or an IDE Stream
 - In some cases, as determined by implementation specific criteria, it may be desirable to implement some other implementation specific action.
- IDE must be enabled in a coordinated way between the two Partner Ports such that both are enabled for IDE prior to either one Transmitting an IDE TLP to the other.
- For Link IDE, software must enable the Upstream Port and then the Downstream Port, while ensuring that no TLPs are transmitted on the Link between these two events.
- For Selective IDE, the Stream must not be used until it has been enabled in both Partner Ports. For cases where one of the Partner Ports is a Root Port and Selective IDE for Configuration Requests is enabled, the other Partner Port must be enabled prior to the Root Port. For other scenarios, the mechanisms to satisfy this requirement are implementation-specific.

6.34 $\uparrow\downarrow$ Unordered IO \downarrow $\uparrow\downarrow$ Unordered I/O \uparrow (UIO) §

Unordered $\uparrow\downarrow$ IO \downarrow $\uparrow\downarrow$ I/O \uparrow (UIO) is an optional capability, intended to address the limitations of the PCI/PCIe fabric-enforced ordering rules. UIO enables fabrics with multiple paths between a source and destination to be supported, and more closely matches the semantics of common IO fabrics (including on-die fabrics). UIO is suitable for all combinations of Requesters and Completers including Host-to-Device, Device-to-Host, and Device-to-Device (P2P).

UIO provides the ability for hardware to maintain full backwards compatibility with the PCI/PCIe producer-consumer model, shifting the responsibility for enforcing the observed ordering from the fabric to the Requester. All UIO Requests have corresponding UIO Completions, and the UIO Completions provide the Requester with the ability (and responsibility) to enforce ordering requirements. UIO can be used only when the entire path from Requester to Completer uses Flit Mode, supports UIO, and has UIO enabled.

6.34.1 UIO Rules §

For the path between a UIO Requester and Completer, UIO must be enabled end-to-end through all intermediate routing elements. For a given operation, the Requester is permitted to select, using ~~↑↓implementation-specific↓~~
~~↑↓implementation specific↑~~ means, between using UIO Requests or non-UIO Requests.

Non-tree topologies and multi-path support are permitted for UIO.

- The mechanism used to enable and manage such topologies is outside the scope of this specification.
- The mechanisms required to avoid deadlocks and loops for such topologies are outside the scope of this specification.
- A tree topology must exist as a subset of any non-tree topologies for enumeration and configuration operations and other non-UIO traffic.
 - Until enabled by implementation specific software, any Links outside of this tree topology must not carry any TLPs other than Messages with Local ($r[2:0] = 100b$) Message routing.

When a VC is configured for UIO, Posted credits are used for UIO Memory Write Requests and Non-Posted credits are used for all other UIO Memory Requests.

A Requester, Completer, or intermediate routing element advertises UIO support through the SVC VC Protocols Supported field in the SVC Resource Capability Register. For UIO functionality that has specific capability and control configuration (see § Section 7.7.9.2 and § Section 7.7.9.3), all UIO-supporting VCs implemented by a Port must have the same capabilities. It is not required for software to enable all implemented UIO-supporting VCs identically.

UIO, as with other PCIe traffic, does not provide protection against address hazarding, that is, the fabric ordering does not consider Request addresses. If such protection is required, it must be implemented outside the scope of this specification.

6.35 MMIO Register Blocks §

MMIO Register Blocks are optional mechanisms for exchanging various types of data structures between system software and a Function. Because all runtime operations are performed using Memory Space, the performance is generally much better than mechanisms such as DOE that use Configuration Space, and the use of Memory Space is better suited to direct assignment of the hardware resources to the appropriate software element.

MMIO Register blocks are discovered using the MMIO Register Block Locator (MRBL) Extended Capability (§ Section 7.9.30). The registers are mapped in Memory Space allocated via a BAR. Each MRBL Locator Register (§ Section 7.9.30.3) entry in the MRBL Extended Capability structure defines the type of MMIO Register Block and the BAR number and the offset within the BAR where these registers are mapped. It is permitted that the BAR contains other items besides the MMIO Register Blocks described in this section.

It is strongly recommended that all accesses to offsets within the MMB Memory Space be naturally aligned. If an access is not naturally aligned, then the request is permitted to be handled as a restricted programming model violation as defined in § Section 2.3.1.

Base 6.4 vs Base 6.3

§ Figure 6-83 portrays the relationship between the MRBL Extended Capability and the MMIO Register Blocks described in this section.

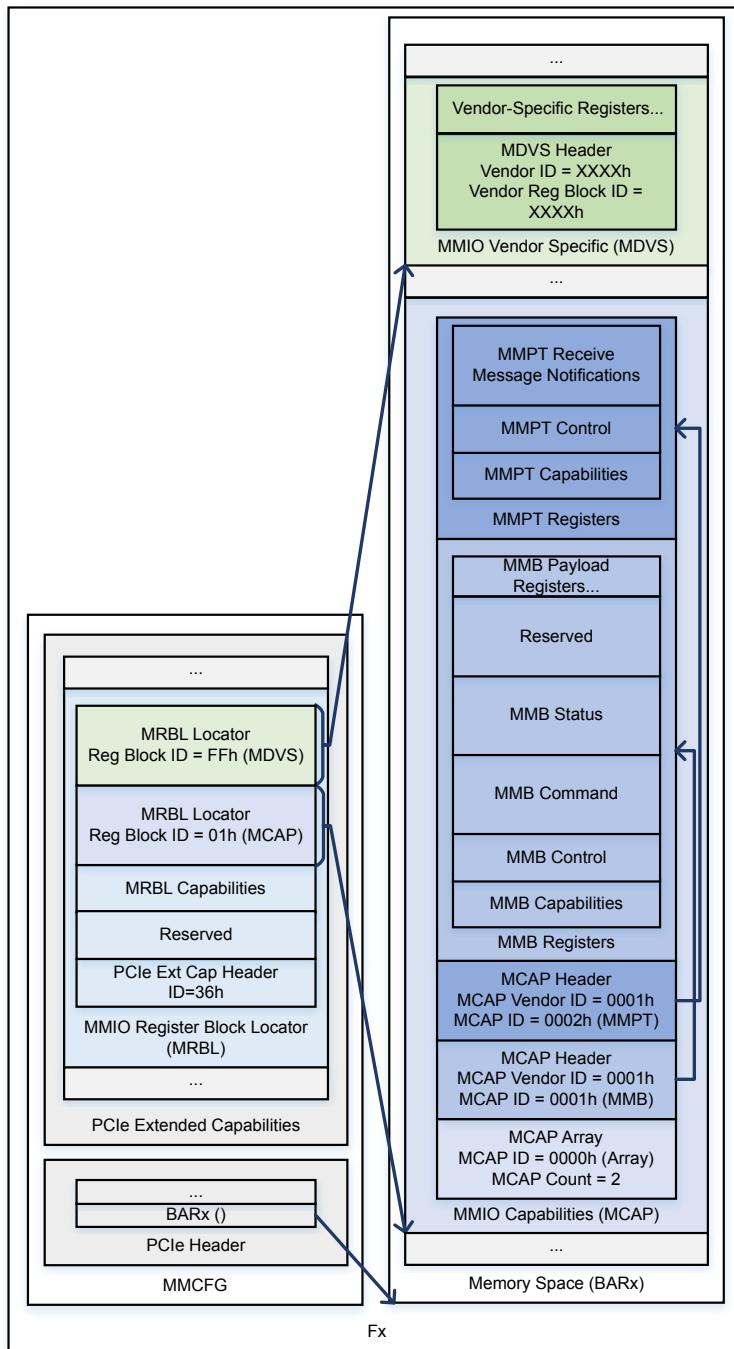


Figure 6-83 MMIO Register Blocks §

6.35.1 MMIO Capabilities Register Block (MCAP) §

The MMIO Capabilities Register Block is an array of capability structures. Each capability in the MCAP array is described by an MCAP Header Register that identifies the specific capability and points to the capability register structure in

Base 6.4 vs Base 6.3

Memory Space. At the beginning of the MCAP Register Block is the MCAP Array Register (see § Section 6.35.1.1) that defines the size of the array followed by a list of MCAP Header Registers (See § Section 6.35.1.2).

The MCAP Register Block is discovered using the MMIO Register Block Locator Extended Capability (MRBL) (see § Section 7.9.30).

§ Figure 6-84 illustrates the MCAP Register Block.

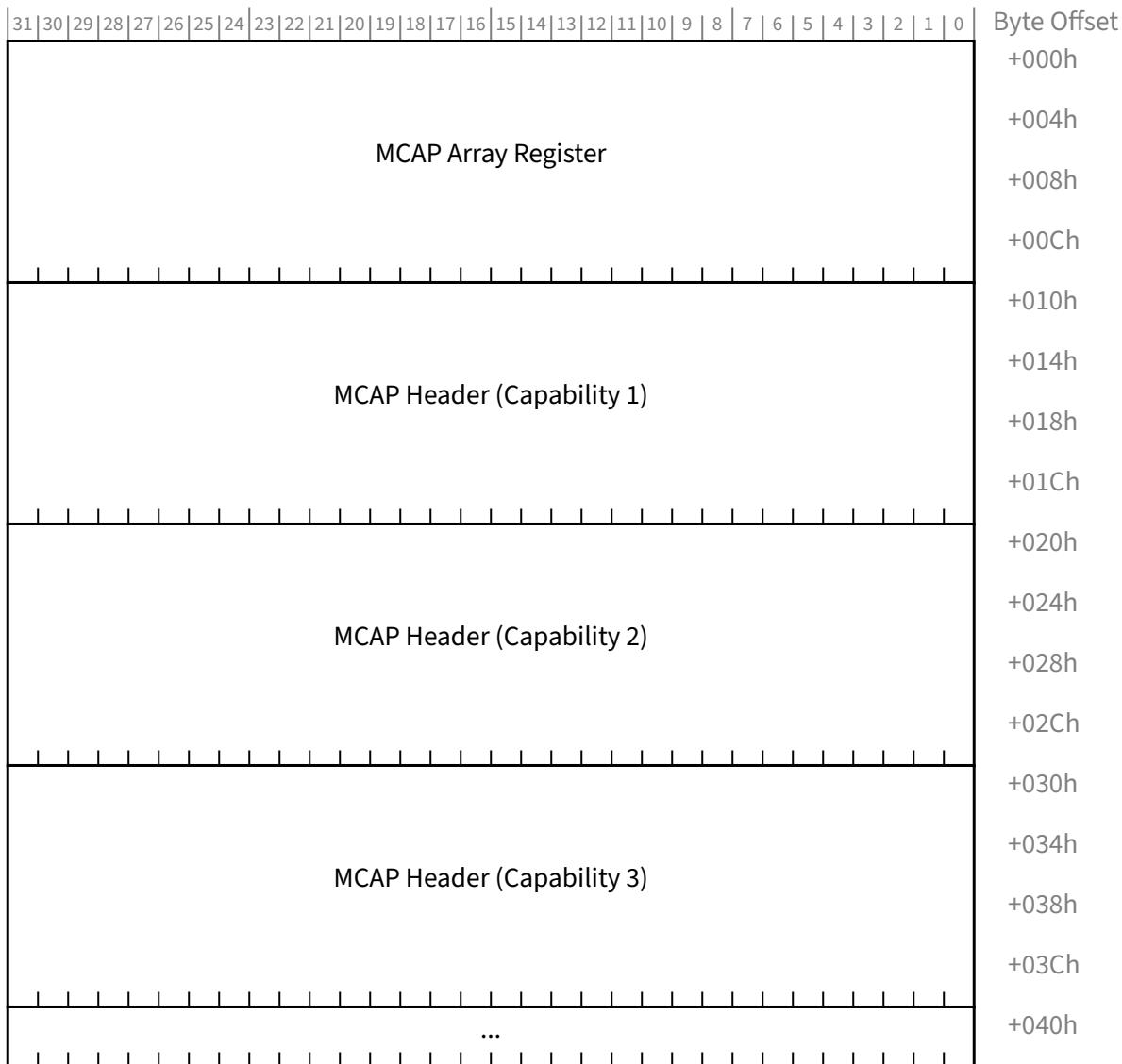


Figure 6-84 MCAP Register Block §

6.35.1.1 MCAP Array Register (Offset 00h) §

§ Figure 6-85 detail allocation of register fields in the MCAP Array Register Block.

Base 6.4 vs Base 6.3

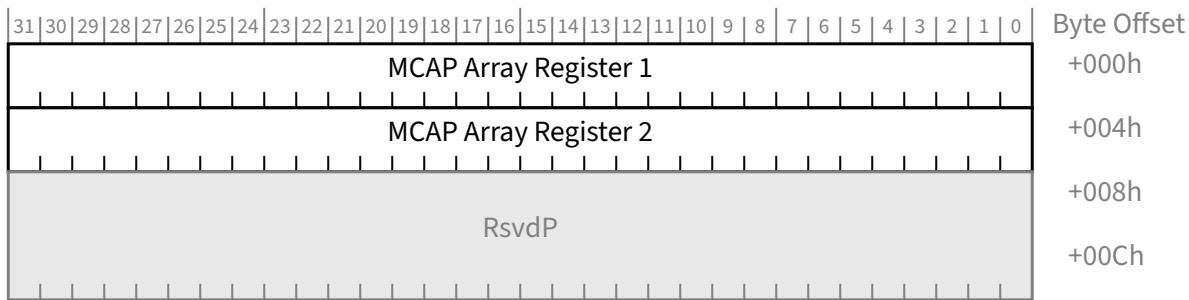


Figure 6-85 MCAP Array Register Block §

§ Figure 6-86 and § Figure 6-87 detail allocation of register fields registers that form the MCAP Array Register Block;
§ Table 6-38 and § Table 6-39 provide the respective bit definitions.

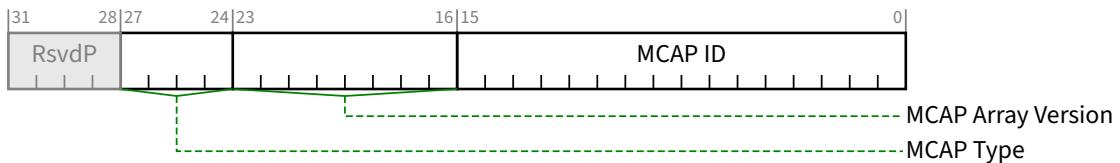


Figure 6-86 MCAP Array Register 1 §

Table 6-38 MCAP Array Register 1 §

| Bit Location | Description | Attributes |
|--------------|---|------------|
| 15:0 | MCAP ID – The MCAP array identifier. For the MCAP Array Register , this field shall be set to 0000h. | RO |
| 23:16 | MCAP Array Version – Defines the format of the MCAP Header Registers (see § Section 6.35.1.2 and § Table 6-40 through § Table 6-43) in the MCAP Register Block . Encodings are: 01h The MCAP Header format is defined in § Figure 6-88 . Others All other encodings are reserved. | RO |
| 27:24 | MCAP Type – Defines the type associated with any type-specific capabilities (MCAP IDs in the range 4000h-7FFFh) in the MCAP Register Block . Encodings are: 0h The type is inferred from the 24-bit value corresponding to the Class Code of the Function. If the Class Code is not associated with any type-specific capabilities, then no type-specific capabilities shall be present. 1h-7h Reserved – Allocated for [CXL]. Others All other encodings are reserved. | RO |

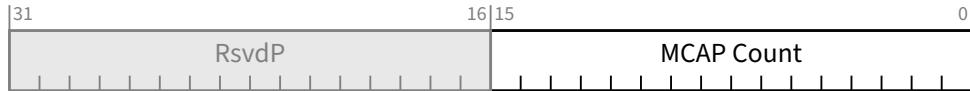


Figure 6-87 MCAP Array Register 2

Table 6-39 MCAP Array Register 2

| Bit Location ↓↑Bit Location↑ | Description | Attributes |
|---------------------------------|--|------------|
| 15:0 | MCAP Count – The number of capabilities in the MCAP Register Block , not including the MCAP Array Register . Each MCAP Header Register (see § Section 6.35.1.2) is contiguous to previous MCAP Header Register . | RO |

6.35.1.2 MCAP Header Register Block (Offset Varies) §

§ Figure 6-88 details allocation of register fields in the MCAP Header Register Block .

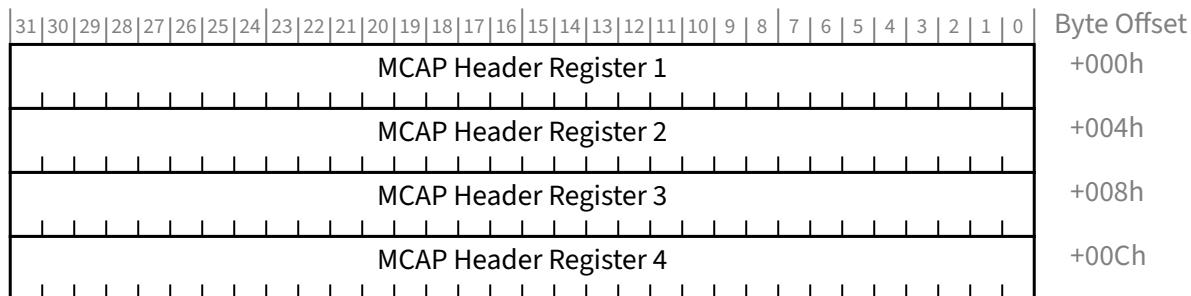


Figure 6-88 MCAP Header Register Block

§ Figure 6-89 through § Figure 6-92 detail allocation of register fields in the MCAP Header Register Block ; § Table 6-40 through § Table 6-43 provides the respective bit definitions.



Figure 6-89 MCAP Header Register 1

Base 6.4 vs Base 6.3

Table 6-40 MCAP Header Register 1 §

| Register Description | | Attributes |
|----------------------|---|------------|
| 15:0 | MCAP ID – The capability identifier. | RO |
| 23:16 | MCAP Version – Defines the version of the capability register structure. The MCAP Version is incremented whenever the capability register structure is extended to add more functionality. Backward compatibility shall be maintained during this process. For all values of n, version n+1 may extend version n by replacing fields that are marked as reserved in version n but must not redefine the meaning of existing fields. Software that was written for a lower version may continue to operate on capability structures with a higher version but will not be able to take advantage of new functionality. If backwards compatibility cannot be maintained, a new MCAP ID shall be created. Each field in a capability register structure is assumed to be introduced in version 1 of that structure unless specified otherwise in the field's definition. | RO |

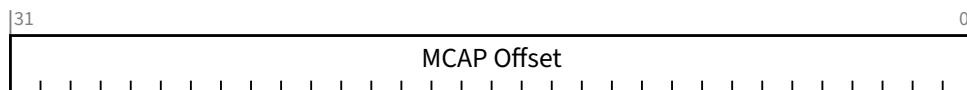


Figure 6-90 MCAP Header Register 2 §

Table 6-41 MCAP Header Register 2 §

| Register Description | | Attributes |
|----------------------|--|------------|
| 31:0 | MCAP Offset – Offset of the capability register structure from the start of the MCAP Register Block in bytes. The offset of performance sensitive MCAPs and security sensitive MCAPs shall be 4 KB aligned within Memory Space. | RO |

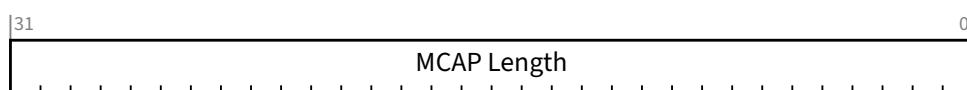


Figure 6-91 MCAP Header Register 3 §

Table 6-42 MCAP Header Register 3 §

| Register Description | | Attributes |
|----------------------|--|------------|
| 31:0 | MCAP Length – Size of the capability register structure in bytes. | RO |

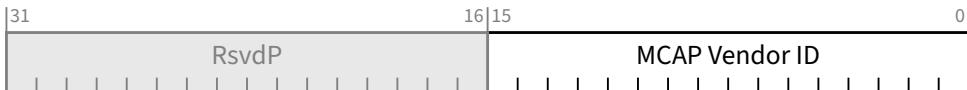


Figure 6-92 MCAP Header Register 4 §

Table 6-43 MCAP Header Register 4 §

| Bit Location ↓↑Bit Location↑ | Register Description | Attributes |
|---------------------------------|---|------------|
| 15:0 | MCAP Vendor ID – The PCI-SIG assigned Vendor ID for the entity that defined the capability register structure. A value of 0000h in this field is reserved for legacy compatibility with existing [CXL] defined register structures only. | RO |

Table 6-44 PCI-SIG Defined MCAP Identifiers (MCAP Vendor ID = 0001h) §

| MCAP Vendor ID | MCAP ID | Description |
|----------------|-------------|---|
| N/A | 0000h | MCAP Array Register – Describes the MCAP Array Register . See § Section 6.35.1.1 . |
| 0001h | 0001h | MMIO Mailbox (MMB) – Describes the MMIO mailbox capability registers (MMB). At most one instance of this register structure can exist per Function. See § Section 6.35.1.3 . |
| 0001h | 0002h | Management Message Passthrough (MMPT) – Describes the registers required for a Function that supports the Management Message Passthrough (MMPT) command set (§ Section 6.36.1). At most one instance of this register structure can exist per Function (See § Section 6.35.1.4.1). |
| 0001h | 0003h-3FFFh | Reserved |
| N/A | 4000h-7FFFh | Type-Specific – Identifies type-specific capabilities associated with the MCAP Type specified in the MCAP Array Register (see § Section 6.35.1.1). |
| 0001h | 8000h-FFFFh | Reserved |

6.35.1.3 MMIO Mailbox Capability (MMB) (Offset: Varies) §

The MMIO Mailbox Capability (MMB) provides the ability to issue a command to a Function.

The MMB interface shall only be used in a single-threaded manner. It is software's responsibility to avoid simultaneous, uncoordinated access to the MMB Registers using techniques such as locking.

The MMB command timeout is 2 seconds. This is the maximum permitted time after the Doorbell is Set in the MMB Control Register (see § Section 6.35.1.3.2.2) for the Function to complete the command, Clear the Doorbell, and optionally signal the Command Ready Interrupt, if configured.

MMB commands do not continue to execute across Conventional Resets. It is not required that FLR results in any type of reset to the internal processing engine for MMB operations, although such behavior is permitted, and may be required or forbidden by specific commands.

The optional MMB Attention Mechanism supports improved power management of MMB implementations, by enabling an MMB instance to temporarily enter an unresponsive state, and providing software, a mechanism to direct the instance back to a responsive state. If MMB Attention Mechanism Support is Set, for backwards compatibility, the default is that the MMB instance must remain in a responsive state. System software is recommended to Set MMB Attention Not Needed when it is acceptable for the MMB instance to enter and stay in a state where it is not immediately available for use. When Clear, the MMB instance must remain in a state where it is ready to respond in a timely way to system software. MMB At Attention , when Set, indicates the MMB interface is presently in a state of readiness. This bit must only be Cleared if MMB Attention Not Needed is Set. It is permitted for this bit to remain Clear for up to 50 ms following the Clearing of MMB Attention Not Needed.

Functions may support sending MSI/MSI-X interrupts to indicate MMB command status. Support for MMB interrupts is enumerated in the MMB Capabilities Register (see § Section 6.35.1.3.2.1) and enabled in the MMB Control Register (see § Section 6.35.1.3.2.2).

Unless specified otherwise in the field definitions for the MMB Registers, each field is present in version 1 and later of these structures. The Function must report the version of these structures in the MCAP Version field of the MCAP Header Register (see § Section 6.35.1.2).

6.35.1.3.1 MMB Operation §

The flow for executing a command is described below. The term “caller” represents the entity submitting the command.

- The caller ensures the Function is ready to accept a new command on the MMB.
 - After a Conventional Reset, the caller ensures the MMB is initialized.
 - The caller polls for MMB Ready to be Set in the MMB Status Register (see § Section 6.35.1.3.2.4).
 - The caller ensures the MMB is in a state of readiness.
 - If MMB Attention Mechanism Support in the MMB Capabilities Register (see § Section 6.35.1.3.2.1) is Set, the caller reads MMB At Attention in the MMB Status Register (see § Section 6.35.1.3.2.4). If Clear:
 - The caller Clears MMB Attention Not Needed in the MMB Control Register (see § Section 6.35.1.3.2.2).
 - The caller either polls for MMB At Attention to be Set in the MMB Status Register (see § Section 6.35.1.3.2.4) or waits for the Command Ready Interrupt if configured.
 - The caller ensures the Function is ready to accept a new command.
 - The caller polls for the Doorbell to be Cleared in the MMB Control Register (see § Section 6.35.1.3.2.2).
- The caller issues a new command.
 - The caller writes the MMB Command Register (see § Section 6.35.1.3.2.3).
 - The caller writes the MMB Payload Registers (see § Section 6.35.1.3.2.5) if the input payload is non-empty.
 - The caller Sets the Doorbell in the MMB Control Register (see § Section 6.35.1.3.2.2).
- The caller waits for the command to complete.
 - The caller either polls for the Doorbell to be Cleared in the MMB Control Register (see § Section 6.35.1.3.2.2) or waits for the Command Ready Interrupt if configured.
 - In case of a command timeout, the caller may attempt to recover the Function by issuing a Conventional reset to the Function.

- The caller retrieves the result of the command.
 - The caller reads the Return Code in the MMB Status Register (see § [Section 6.35.1.3.2.4](#)).
 - The caller reads the MMB Payload Length in the MMB Command Register (see § [Section 6.35.1.3.2.3](#)). If non-zero, the caller reads the MMB Payload Registers (see § [Section 6.35.1.3.2.5](#)) to retrieve the output payload.

6.35.1.3.2 MMB Registers §

§ Figure 6-93 illustrates the MMB Registers structure.

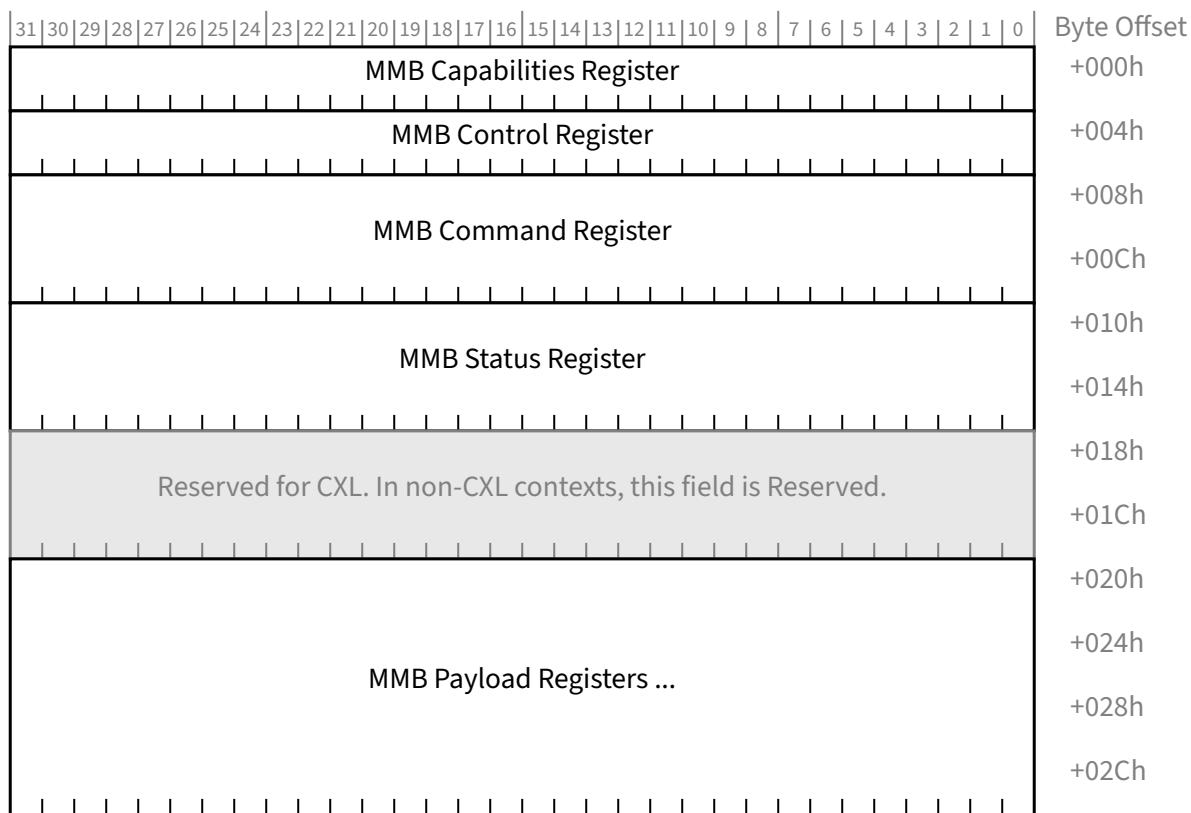


Figure 6-93 MMB Registers §

6.35.1.3.2.1 MMB Capabilities Register (Offset 00h) §

§ Figure 6-94 details allocation of register fields in the MMB Capabilities Register ; § [Table 6-45](#) provides the respective bit definitions.

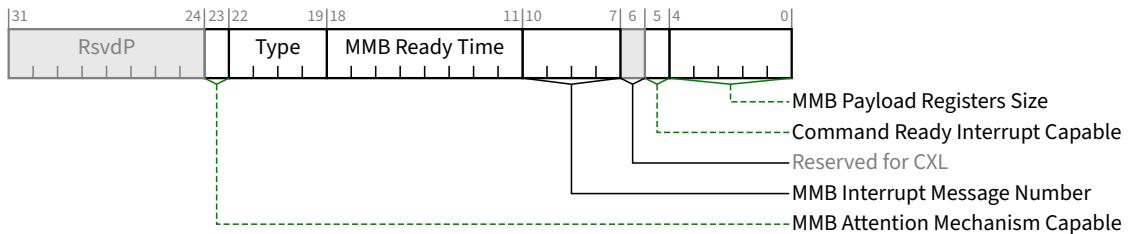


Figure 6-94 MMB Capabilities Register §

Table 6-45 MMB Capabilities Register §

| Bit Location ↑↓↑↓↑↑ | Register Description | Attributes |
|------------------------|--|---------------|
| 4:0 | MMB Payload Registers Size – Size of the MMB Payload Registers (see § Section 6.35.1.3.2.5) in bytes, expressed as 2^n . The minimum size is 256 bytes (n=8) and the maximum size is 1 MB (n=20). | HWInit |
| 5 | Command Ready Interrupt Capable – This field indicates if the MMB supports signaling an MSI/MSI-X interrupt when the Doorbell in the MMB Control Register (see § Section 6.35.1.3.2.2) transitions from Set to Clear or MMB At Attention in the MMB Status Register (see § Section 6.35.1.3.2.4) transitions from Clear to Set. 0 Not supported 1 Supported | HWInit |
| 6 | Reserved for CXL – This field is assigned for use by [CXL]. In non-CXL contexts, this field is Reserved. | RsvdP |
| 10:7 | MMB Interrupt Message Number – When MSI/MSI-X is implemented, this field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with this MMB instance. For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control register for MSI. For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry shall be one of the first 16 entries even if the Function implements more than 16 entries. The value in this field shall be within the range configured by system software to the Function. For a given MSI-X implementation, the entry shall remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field shall indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field indicates the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined. | RO / RsvdP |
| 18:11 | MMB Ready Time – This field indicates the maximum amount of time in seconds after a Conventional Reset for MMB Ready in the MMB Status Register (see § Section 6.35.1.3.2.4) to become Set. A value of 0 indicates 2 seconds. | HWInit |
| 22:19 | Type – This field identifies the type-specific commands supported by the MMB. 0h The type is inferred from the 24-bit value corresponding to the Class Code of the Function. If the Class Code is not associated with any type-specific commands, then no type-specific commands are present. 1h-7h Reserved – Allocated for [CXL]. Others All other encodings are reserved. | HWInit |

| Bit Location | Register Description | Attributes |
|---------------------|---|---------------|
| 23 | MMB Attention Mechanism Capable – This field indicates if the MMB supports the optional MMB Attention Mechanism. 0 Not supported 1 Supported | <u>HWInit</u> |

6.35.1.3.2.2 MMB Control Register (Offset 04h) §

§ Figure 6-95 details allocation of register fields in the MMB Control Register ; § Table 6-46 provides the respective bit definitions.

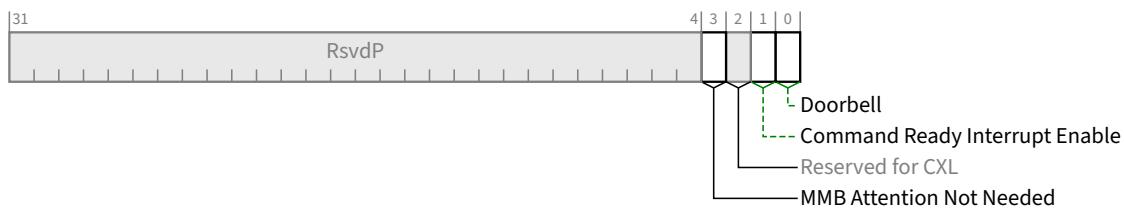


Figure 6-95 MMB Control Register §

Table 6-46 MMB Control Register §

| Bit Location | Register Description | Attributes |
|---------------------|--|-------------------|
| 0 | Doorbell – When MMB Ready is Set, this bit, when Clear, indicates that the Function is ready to accept a new command. Set by the caller to notify the Function that the command inputs are ready. Read-only when Set. Cleared by the Function when the command completes. 0 Ready to accept a command 1 Busy executing a command or initializing Default value of this field is 1b. Cleared by the Function when <u>MMB Ready</u> is Set in the <u>MMB Status Register</u> . | <u>RW</u> |
| 1 | Command Ready Interrupt Enable – When <u>Command Ready Interrupt Capable</u> is Set, this bit, when Set, enables the MMB to signal an MSI/MSI-X interrupt when the <u>Doorbell</u> transitions from Set to Cleared or <u>MMB At Attention</u> in the <u>MMB Status Register</u> (see § <u>Section 6.35.1.3.2.4</u>) transitions from Clear to Set. Read-only when the <u>Doorbell</u> is Set. When <u>Command Ready Interrupt Capable</u> is Clear, this bit is permitted to be <u>RsvdP</u> . 0 Disabled 1 Enabled Default value of this field is 0b. | <u>RW / RsvdP</u> |
| 2 | Reserved for CXL – This field is assigned for use by [CXL]. In non-CXL contexts, this field is Reserved. | <u>RsvdP</u> |
| 3 | MMB Attention Not Needed – If <u>MMB Attention Mechanism Capable</u> is Set, this bit, when Set, enables the MMB to enter and stay in a state where it is not immediately available for use. When Clear the MMB must remain in a responsive state. | <u>RW / RsvdP</u> |

| ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|------------------------------------|---|------------|
| | <p>When MMB Attention Mechanism Support is Clear, this bit is permitted to be RsvdP .</p> <p>0 Attention needed 1 Attention not needed</p> <p>Default value of this bit is 0b.</p> | |

6.35.1.3.2.3 MMB Command Register (Offset 08h) §

The MMB Command Register shall only be used by the caller when the Doorbell in the MMB Control Register (see § Section 6.35.1.3.2.2) is Cleared.

§ Figure 6-96 details allocation of register fields in the MMB Command Register ; § Table 6-47 provides the respective bit definitions.

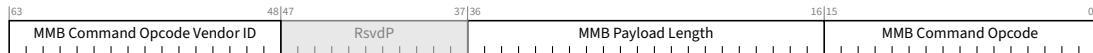


Figure 6-96 MMB Command Register §

Table 6-47 MMB Command Register §

| ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|------------------------------------|--|------------|
| 15:0 | <p>MMB Command Opcode – The command identifier.</p> <p>Default value of this field is 0000h.</p> | RW |
| 36:16 | <p>MMB Payload Length – The size of the data in the MMB Payload Registers (see § Section 6.35.1.3.2.5) in bytes. Valid values for this field are less than or equal to the MMB Payload Registers Size specified in the MMB Capabilities Register (see § Section 6.35.1.3.2.1). Written by the caller to provide the command input payload size to the Function prior to setting the Doorbell in the MMB Control Register (see § Section 6.35.1.3.2.2). Values specified by the caller that are greater than the MMB Payload Registers Size result in an error of Invalid Payload Length being returned in the MMB Return Code field. Written by the Function to provide the command output payload size to the caller when the Doorbell is Cleared in the MMB Control Register (see § Section 6.35.1.3.2.2).</p> <p>Default value of this field is 0000h.</p> | RW |
| 63:48 | <p>MMB Command Opcode Vendor ID – The PCI-SIG assigned Vendor ID for the entity that defined the MMB Command Opcode. A value of 0000h in this field is reserved for legacy compatibility with existing [CXL] defined commands only.</p> <p>Default value of this field is 0000h.</p> | RW |

IMPLEMENTATION NOTE: MMB COMMAND OPCODE VENDOR ID LEGACY COMPATIBILITY §

For legacy compatibility with OS software, platform firmware should clear the MMB Command Opcode Vendor ID to 0000h prior to OS hand off.

6.35.1.3.2.4 MMB Status Register (Offset 10h) §

Reports information about the state of the MMB and the last command executed since Conventional Reset.

§ Figure 6-97 details allocation of register fields in the MMB Status Register ; § Table 6-48 provides the respective bit definitions.

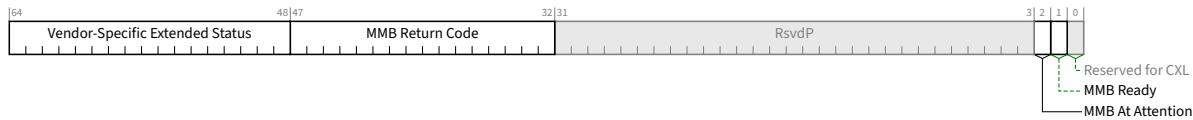


Figure 6-97 MMB Status Register §

Table 6-48 MMB Status Register §

| Bit Location ↓↑Bit Location↓ ↑↑Bit Location↑ | Register Description | Attributes |
|--|---|---------------|
| 0 | Reserved for CXL – This field is assigned for use by [CXL]. In non-CXL contexts, this field is Reserved. | RsvdP |
| 1 | MMB Ready – This bit, when Set, indicates that the Function is ready to accept commands through the MMB interface. Functions that report a non-zero MMB Ready Time shall Set this bit after a Conventional Reset within the time reported in the MMB Ready Time field and it shall remain Set until the next Conventional Reset, or the Function encounters an error that prevents any MMB communication. 0 Not ready 1 Ready | RO |
| 2 | MMB At Attention – When MMB Attention Mechanism Capable is Set, this bit, when Set, indicates the MMB interface is presently in a state of readiness. The transition of this bit from Clear to Set triggers a Command Ready interrupt if Command Ready Interrupt Enable is Set. When MMB Attention Mechanism Support is Clear, this bit is RsvdP. This bit is Set by the Function when MMB Ready is Set. 0 Not at Attention 1 At Attention | RO / RsvdP |
| 47:32 | MMB Return Code – The result of the command. Only valid after the Doorbell is Cleared in the MMB Control Register (see § Section 6.35.1.3.2.2) after a command completes. | RO |
| 64:48 | Vendor-Specific Extended Status – The Vendor-Specific extended status information. Only valid after the Doorbell is Cleared in the MMB Control Register (see § Section 6.35.1.3.2.2) after a command completes. | RO |

6.35.1.3.2.4.1 MMB Command Return Codes §

Command Return Codes are associated with the entity that defined the command as indicated by the MMB Command Opcode Vendor ID in the MMB Command Register.

The MMB Command Return Code is Set by the Function to indicate to the Caller the result of a command. If more than one MMB Command Return Code applies, the Function chooses which one to return.

In general, retries are not recommended for commands that return an error except when indicated in the return code definition.

PCI-SIG defined command return codes are only valid for PCI-SIG defined commands.

Table 6-49 MMB PCI-SIG Defined Command Return Codes (Vendor ID = 0001h) §

| Value | Description |
|-------------|---|
| 0000h | Success : The command successfully completed. |
| 0001h | Reserved |
| 0002h | Invalid Input : A command input was invalid. |
| 0003h | Unsupported : The MMB Command Opcode is not supported. |
| 0004h | Internal Error : The command was not completed because of an internal error. |
| 0005h | Retry Required : The command was not completed because of a temporary error. An optional single retry may resolve the issue. |
| 0006h-7FFFh | Reserved : Allocated for [CXL]. |
| 8000h | Invalid Payload Length : The input payload length specified for the command is not valid. |
| 8001h | Unsupported Management Message : The MMPT Message Type is not supported. |
| 8002h | Transfer in Progress : Only one transfer can occur at a time. Complete or abort the current data transfer before starting a new one. |
| 8003h | Transfer Out of Order : The transfer was aborted because the data was transferred out of order. |
| 8004h | Transfer Aborted : The data transfer was aborted and must be restarted from the beginning. |
| 8005h-FFFFh | Reserved |

6.35.1.3.2.5 MMB Payload Registers (Offset 20h) §

The MMB Payload Registers must consist of 256 Bytes – 1 MB, as shown in § Figure 6-98 . The size of the MMB Payload Registers is reported in the MMB Capabilities Register (see § Section 6.35.1.3.2.1). Any data beyond the size specified in the MMB Capabilities Register shall be ignored by the caller and the Function.

The MMB Payload Registers are written by the caller to provide the command input payload to the Function prior to Setting the Doorbell in the MMB Control Register (see § Section 6.35.1.3.2.2). They are written by the Function to provide the command output payload back to the caller prior to Clearing the Doorbell .

These registers must only be used by the caller when the Doorbell in the MMB Control Register (see § Section 6.35.1.3.2.2) is Clear.

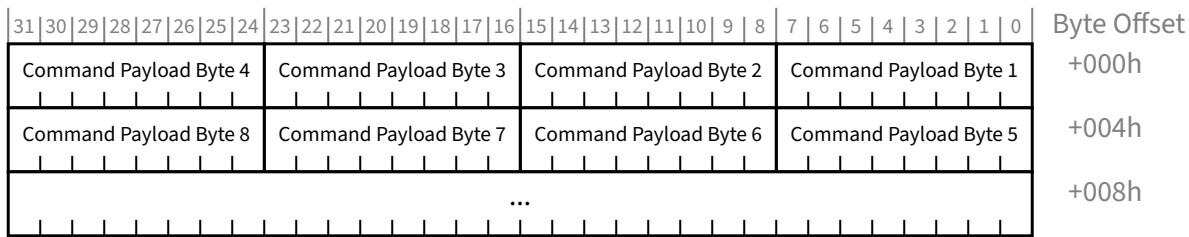


Figure 6-98 MMB Payload Registers

6.35.1.4 Management Message Passthrough (MMPT) Capability (Offset: Varies)

The Management Message Passthrough (MMPT) Capability is required for Functions that support the MMPT command set. Refer to the MMPT command interface for usage of this capability ([§ Section 6.36.1](#)).

Unless specified otherwise in the field definitions, each field is present in version 1 and later of this structure. The Function shall report the version of this structure in the [MCAP Version](#) field of the MCAP Header Register .

6.35.1.4.1 MMPT Registers

§ Figure 6-99 illustrates the MMPT Register structure.

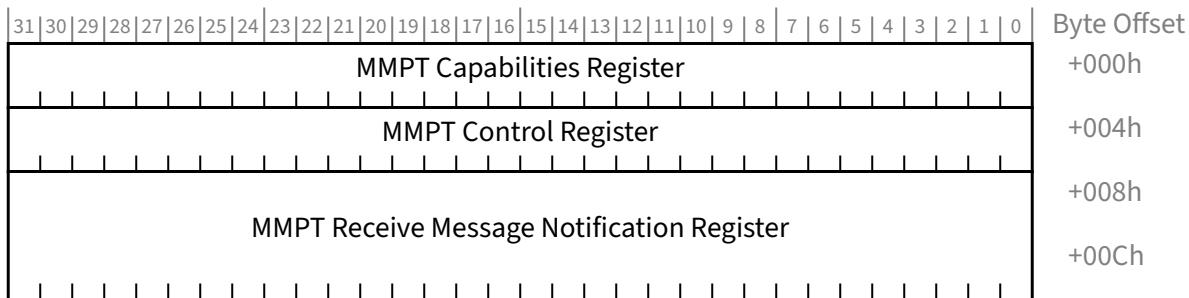


Figure 6-99 MMPT Registers

6.35.1.4.1.1 MMPT Capabilities Register (Offset 00h)

§ Figure 6-100 details allocation of register fields in the MMPT Capabilities Register ; § Table 6-50 provides the respective bit definitions.

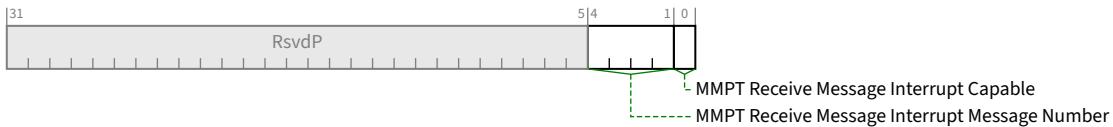


Figure 6-100 MMPT Capabilities Register §

Table 6-50 MMPT Capabilities Register §

| Bit Location ↓↑Bit Location↓ ↑↑Bit Location↑ | Register Description | Attributes |
|--|---|------------|
| 0 | <p>MMPT Receive Message Interrupt Capable – This field indicates if the Function supports signaling an MSI/MSI-X interrupt when MMPT Receive Message Ready transitions from Clear to Set.</p> <p>0 Not supported</p> <p>1 Supported</p> | RO |
| 4:1 | <p>MMPT Receive Message Interrupt Message Number – When MSI/MSI-X is implemented, this field indicates which MSI/MSI-X vector is used for the interrupt message generated when MMPT Receive Message Ready transitions from Clear to Set.</p> <p>For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the <u>Multiple Message Enable</u> field in the <u>Message Control</u> register for MSI.</p> <p>For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry shall be one of the first 16 entries even if the Function implements more than 16 entries. The value in this field shall be within the range configured by system software to the device. For a given MSI-X implementation, the entry shall remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field shall indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field indicates the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.</p> | RO / RsvdP |

6.35.1.4.1.2 MMPT Control Register (Offset 04h) §

§ Figure 6-101 details allocation of register fields in the MMPT Control Register; § Table 6-51 provides the respective bit definitions.

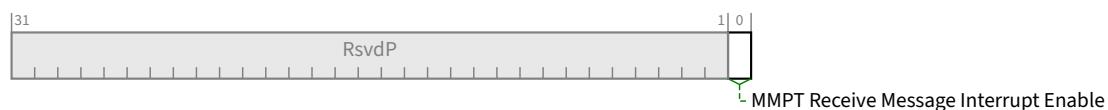


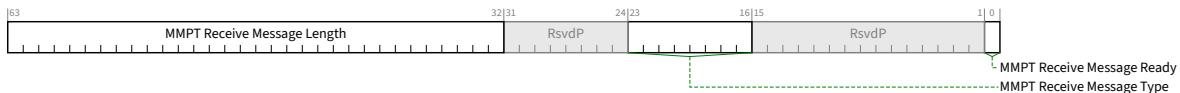
Figure 6-101 MMPT Control Register §

Table 6-51 MMPT Control Register

| Register Description | | Attributes |
|----------------------|--|------------|
| 0 | <p>MMPT Receive Message Interrupt Enable – When MMPT Receive Message Interrupt Capable is Set, this bit, when Set, enables the Function to signal an MSI/MSI-X interrupt when MMPT Receive Message Ready transitions from Clear to Set. When MMPT Receive Message Interrupt Capable is Clear, this bit is permitted to be RsvdP.</p> <p>0 Disabled 1 Enabled</p> <p>Default value of this field is 0b.</p> | RW / RsvdP |

6.35.1.4.1.3 MMPT Receive Message Notification Register (Offset 08h)

§ Figure 6-102 details allocation of register fields in the MMPT Receive Message Notification Register ; § Table 6-52 provides the respective bit definitions.

*Figure 6-102 MMPT Receive Message Notification Register**Table 6-52 MMPT Receive Message Notification Register*

| Register Description | | Attributes |
|----------------------|---|------------|
| 0 | MMPT Receive Message Ready – When Set, a new management message is ready to be transferred from the Function to the host using the MMPT Receive Message command (see § Section 6.36.1.2). | RO |
| 23:16 | MMPT Receive Message Type – This field indicates the nature and format of the management message ready to be transferred from the Function to the host using the MMPT Receive Message command (see § Section 6.36.1.2). Encodings for this field are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved. | RO |
| 63:32 | MMPT Receive Message Length – This field indicates the length of the management message in bytes ready to be transferred from the Function to the host using the MMPT Receive Message command (see § Section 6.36.1.2). | RO |

6.35.2 MMIO Designated Vendor-Specific Register Block (MDVS)

The MMIO Designated Vendor-Specific Register Block (MDVS) allows a Vendor-Specific Memory Space register block to be discovered by utilizing the MRBL Extended Capability (§ Section 7.9.30). The format of the MDVS Register Block starts with the MDVS Register Block Header Register (see § Section 6.35.2.1 and § Section 6.35.3) as illustrated in § Figure 6-103 . The remainder of the MDVS Register Block is vendor defined. A single Function is permitted to implement more than one in a MDVS Register Block .

Base 6.4 vs Base 6.3

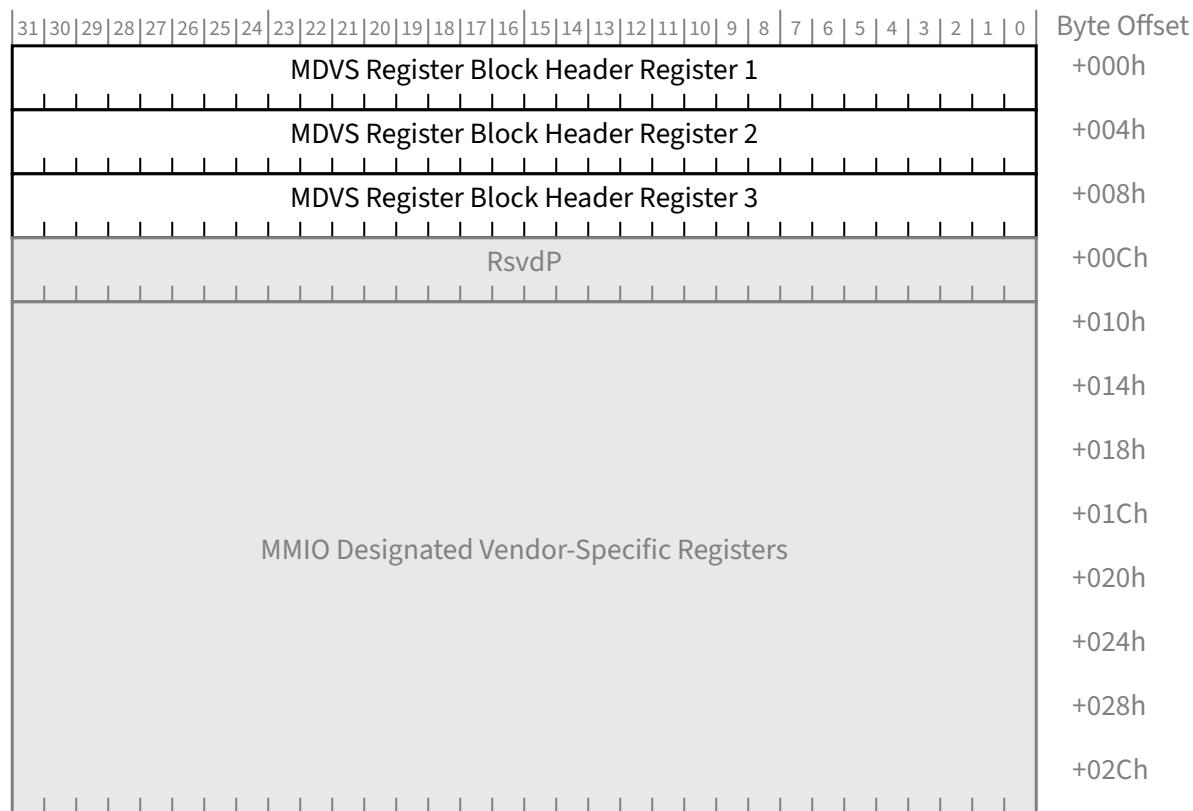


Figure 6-103 MDVS Register Block §

6.35.2.1 MDVS Register Block Header Register 1 (Offset 00h) §

§ Figure 6-104 details allocation of register fields in the MDVS Register Block Header Register 1 ; § Table 6-53 provides the respective bit definitions.

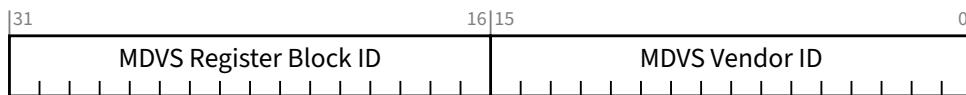


Figure 6-104 MDVS Register Block Header Register 1 §

Table 6-53 MDVS Register Block Header Register 1 §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | MDVS Vendor ID – The PCI-SIG assigned Vendor ID for the organization that defined the layout and controls the specification for this register block. | RO |

| <small>↑↓Bit Location↓ ↑↑Bit Location↑</small> | Register Description | Attributes |
|--|--|------------|
| 31:16 | MDVS Register Block ID – Value defined by the Vendor ID in bits 15:0 that indicates the nature and format of the Vendor-Specific registers. | RO |

6.35.3 MDVS Register Block Header Register 2 (Offset 04h) §

§ Figure 6-105 details allocation of register fields in the MDVS Register Block Header Register 2 ; § Table 6-54 provides the respective bit definitions.

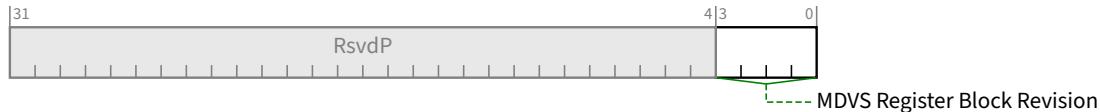


Figure 6-105 MDVS Register Block Header Register 2 §

Table 6-54 MDVS Register Block Header Register 2 §

| <small>↑↓Bit Location↓ ↑↑Bit Location↑</small> | Register Description | Attributes |
|--|--|------------|
| 3:0 | MDVS Register Block Revision – Version number defined by the Vendor ID in bits 15:0 that indicates the version of the register block. | RO |

6.35.4 MDVS Register Block Header Register 3 (Offset 08h) §

§ Figure 6-106 details allocation of register fields in the MDVS Register Block Header Register 3 ; § Table 6-55 provides the respective bit definitions.



Figure 6-106 MDVS Register Block Header Register 3 §

Table 6-55 MDVS Register Block Header Register 3 §

| <small>↑↓Bit Location↓ ↑↑Bit Location↑</small> | Register Description | Attributes |
|--|--|------------|
| 31:0 | MDVS Register Block Length – The number of bytes in the register block, including the MDVS Register Block Header and the Vendor-Specific registers. | RO |

6.36 MMB Command Interface §

MMB commands are identified by a MMB Command Opcode. MMB Command Opcodes also provide an implicit version number, which means a command's definition shall not change in an incompatible way in future revisions of this specification. Instead, if an incompatible change is required, the specification defining the change shall define a new MMB Command Opcode for the changed command. Commands may evolve by defining new fields in areas of the payload definitions that were originally defined as Reserved, but only in a way where software written using the earlier definition will continue to work correctly, and software written to the new definition can use the 0 value or the payload size to detect Function components that do not support the new field. This implicit versioning allows software to be written with the understanding that a MMB Command Opcode shall only evolve by adding backward compatible changes.

Table 6-56 PCI-SIG Defined MMB Command Opcodes (Vendor ID = 0001h) §

| MMB Command Opcode | | | | | Input Payload Size (Bytes) (Note 1) | Output Payload Size (Bytes) (Note 1) |
|------------------------|---------------------------------------|-------------------|--|-----------------|-------------------------------------|--------------------------------------|
| Command Set Bits[15:8] | | Command Bits[7:0] | | Combined Opcode | | |
| 00h | Reserved | 00h-FFh | Reserved | 0000h-00FFh | N/A | N/A |
| 01h | Management Message Passthrough (MMPT) | 00h | Send Message (see § Section 6.36.1.1) | 0100h | 10h+ | 00h+ |
| | | 01h | Receive Message (see § Section 6.36.1.2) | 0101h | 10h | 00h+ |
| | | 02h-FFh | Reserved | 0102h-01FFh | N/A | N/A |
| 02h-FFh | Reserved | 00h-FFh | Reserved | 0200h-FFFFh | N/A | N/A |

Notes:

1. Indicates the minimum payload size for a successful completion. Commands with variable payloads sizes are marked with '+'. Actual valid bytes in the output payload are indicated by the MMB Payload Length field in the MMB Command Register (see § Section 6.35.1.3.2.3).

6.36.1 Management Message Passthrough (MMPT) §

The optional Management Message Passthrough (MMPT) command set provides an interface to tunnel management messages defined in other industry standard specifications between the host and the Function using the MMB Command Interface. This enables OS standard class drivers to utilize the same management messages used by the BMC, thus making in-band and out-of-band management more consistent.

Management messages may be initiated from the host to the Function using the Send Message command (see § Section 6.36.1.1) or asynchronously from the Function to the host using the Receive Message command (see § Section 6.36.1.2). The Function notifies the host that a new management message is ready to be retrieved using the Receive Message command (see § Section 6.36.1.2) by setting the MMPT Receive Message Ready bit in the MMPT Receive Message Notification Register (see § Section 6.35.1.4.1.3). If more than one management message is ready to be transferred from the Function to the host, it is the Function's responsibility to queue up the management messages until the host retrieves them.

Management messages may be larger than the MMB Payload Registers Size requiring the management message to be transferred in parts. If a management message is transferred in its entirety, the caller makes one call to Send or Receive Message with Action = Full Data Transfer. The Offset field is not used and shall be ignored. If a management message is transferred in parts, the caller makes one call to Send or Receive Message with Action = Initiate Data Transfer, zero or more calls with Action = Continue Data Transfer, and one call with Action = Finish Data Transfer or Abort Data Transfer. The management message parts shall be transferred in ascending order based on the Offset value, and the Function shall return the Transfer Out of Order return code if the management message parts are not transferred in ascending order. Back-to-back retransmission of any management message part is permitted during a transfer. A Send or Receive Message command with Action = Abort Data Transfer shall be supported for management messages that can be transferred in parts. An attempt to call Send or Receive Message with Action = Abort Data Transfer for a message whose data has been fully transferred shall fail with Invalid Input.

If the management message transfer is interrupted by a Conventional Reset, error condition, or MMB command-specific reason, the management message transfer shall be aborted and the Send or Receive Message command shall return Transfer Aborted. If a management message transfer is aborted prior to the entire management message data being transferred, the Function shall require the management message transfer to be started from the beginning of the message data.

6.36.1.1 MMPT Send Message (Opcode 0100h) §

Transfer all or part of a management message from the host to the Function. If a management message is transferred in parts, the entire management data shall be transferred or aborted prior to transferring a new management message. Otherwise, the function shall return Invalid Input.

Possible MMB Command Return Code values (see § Section 6.35.1.3.2.4.1):

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Unsupported
- Unsupported Management Message
- Transfer Out of Order
- Transfer in Progress
- Transfer Aborted

Table 6-57 MMPT Send Message Input Payload §

| Byte Offset | Length in Bytes | Description |
|-------------|-----------------|--|
| 00h | 01h | MMPT Message Type – This field indicates the nature and format of the Message Data. Encodings for this field are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved. |
| 01h | 03h | Reserved |
| 04h | 04h | Flags DW Bits[2:0] Action Specifics – Specifies the stage of the Message Data transfer. |

| Byte Offset | Length in Bytes | Description | |
|-------------|-----------------|--|-------------------------------|
| | | 000b | Full Data Transfer |
| | | 001b | Initiate Data Transfer |
| | | 010b | Continue Data Transfer |
| | | 011b | Finish Data Transfer |
| | | 100b | Abort Data Transfer |
| | | Others | All other values are reserved |
| | | Bits[31:3] | Reserved |
| 08h | 04h | Offset – The Byte Offset in the Message Data. | |
| 0Ch | 04h | Reserved | |
| 10h | variable | Message Data – The management message data, formatted according to the MMPT Message Type specification. | |

Table 6-58 MMPT Send Message Output Payload §

| Byte Offset | Length in Bytes | Description | |
|-------------|-----------------|--|--|
| 00h | Varies | Message Data – The management message response, formatted according to the MMPT Message Type specification. | |

6.36.1.1.1 MMPT Send Message Operation §

The flow for transferring a management message from the host to the Function is described below. Total input payload size refers to the [MMPT Send Message Input Payload](#) structure (see § Table 6-57) including the full Message Data.

- The host transfers the management message to the Function using the Send Message command.
 - If the total input payload size is less than or equal to the [MMB Payload Registers Size](#), the host issues the Send Message Command with Action = Full Data Transfer.
 - If the total input payload size is greater than the [MMB Payload Registers Size](#), the host transfers the message to the Function in ordered parts where each part fits in the [MMB Payload Registers](#) (§ Section 6.35.1.3.2.5). If any part fails to transfer, the host can retransmit that part before sending the next part or abort the transfer and start over.
 - The host issues the Send Message Command with Action = Initiate Data Transfer.
 - Depending on the size of the Message Data, the host issues zero or more Send Message Commands with Action = Continue Data Transfer.
 - The host issues the Send Message Command with Action = Finish Data Transfer.
- The Function may optionally send a response message back to the host.
 - If the response message size is less than or equal to the [MMB Payload Registers Size](#) and the response message data is available when the Send Message command completes, the Function returns the response message in the output payload.
 - Otherwise, the Function sends the response message using the asynchronous Receive Message flow.

6.36.1.2 MMPT Receive Message (Opcode 0101h) §

Transfer all or part of a management message from the Function to the host.

Possible MMB Command Return Code values (§ Section 6.35.1.3.2.4.1):

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Unsupported
- Unsupported Management Message
- Transfer Out of Order
- Transfer in Progress
- Transfer Aborted

Table 6-59 MMPT Receive Message Input Payload §

| Byte Offset | Length in Bytes | Description | |
|-------------|-----------------|---|-----------------------------------|
| 00h | 01h | MMPT Message Type – This field indicates the nature and format of the Message Data. Encodings for this field are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved. | |
| 01h | 03h | Reserved | |
| 04h | 04h | Flags DW : Bits[2:0] Action Specifics – Specifies the stage of the Message Data transfer. 000b Full Data Transfer 001b Initiate Data Transfer 010b Continue Data Transfer 011b Finish Data Transfer 100b Abort Data Transfer Others All other values are reserved | Bits[31:3] Reserved |
| 08h | 04h | Offset – The Byte Offset in the message data to return in the output payload. | |
| 0Ch | 04h | Message Length – The Length in Bytes of the message data to return in the output payload. | |

Table 6-60 MMPT Receive Message Output Payload §

| Byte Offset | Length in Bytes | Description | |
|-------------|-----------------|--|--|
| 00h | Varies | Message Data – The management message data, formatted according to the MMPT Message Type specification. | |

6.36.1.2.1 MMPT Receive Message Operation §

The flow for transferring a management message asynchronously from the Function to the Host is described below.

- The Function indicates to the host that a new management message is ready to be retrieved.
 - The Function writes the MMPT Receive Message Notification Register (§ Section 6.35.1.4.1.3) and Sets MMPT Receive Message Ready .
 - The host either polls for MMPT Receive Message Ready to be Set or waits for the MMPT Receive Message Interrupt if configured.
- The host transfers the management message from the Function using the Receive Message command.
 - If the MMPT Receive Message Length is less than or equal to the MMB Payload Registers Size , the host issues the Receive Message Command with Action = Full Data Transfer.
 - If the MMPT Receive Message Length is greater than the MMB Payload Registers Size , the host transfers the management message from the Function in ordered parts where each part fits in the MMB Payload Registers (see § Section 6.35.1.3.2.5). If any part fails to transfer, the host can retransmit that part before receiving the next part or abort the transfer and start over.
 - The host issues the Receive Message Command with Action = Initiate Data Transfer.
 - Depending on MMPT Receive Message Length , the host issues zero or more Receive Message Commands with Action = Continue Data Transfer.
 - The host issues the Receive Message Command with Action = Finish Data Transfer.
- The Function Clears the MMPT Receive Message Notification Register (see § Section 6.35.1.4.1.3).

6.37 Debug Over Link §

As complexity of interconnects and SoCs/chiplets increases, the complexity of diagnosing and resolving issues relating to these increases as well. The ability to transmit real-time debug information is very useful for identifying and addressing problems that may arise during operation. Debug mechanisms, such as the ones described in this section, are tools that provide the visibility needed to ease the failure debug process.

6.37.1 NOP Flit §

The NOP Flit debug mechanism is a method for transmitting debug information over the PCI Express link. It utilizes the NOP.Debug and NOP.Vendor Flit types (see § Section 4.2.3.4.3.2 and § Section 4.2.3.4.3.3), which are inherently non-intrusive and conform to existing PCI Express flit transmission rules, to deliver the debug information across the link. This mechanism can be particularly useful for capturing real-time link information, which is vital for debugging transient issues that are not easily accessible or visible after their occurrence.

Both NOP.Debug and NOP.Vendor Flits can provide visibility of internal state information to an observer. Internal states could be related to the PCI Express Link used for sending these NOP Flit types or it could be something unrelated to the PCI Express Link.

Possible Observers include implementation specific entities in the receiving Port and external Protocol/Logic Analyzers, allowing for real-time analysis of link behavior. This immediate access to link information can be important for diagnosing issues that may only be present for a brief period, allowing that critical debug data is not missed or stale.

NOP Flits are transmitted by the Physical Layer, functioning independently from the upper protocol layers with respect to their transmission. While the content encapsulated within these NOP Flits may be derived from or informed by the

upper layers, such as the Data Link or Transaction Layers, the actual process of sending these Flits is managed at the Physical Layer level. This separation ensures that the debug information can be transmitted even in scenarios where the upper layers may not be fully operational or are in a state of initialization.

NOP.Debug Flits can be transmitted periodically during normal operation as a proactive measure serving as checkpoints, as an early indicator preceding potential error conditions, in direct response to error conditions that have been detected, or manually triggered. Note: NOP.Vendor Flits can be used in a similar manner.

Possible usages of NOP.Debug Flits:

1. Periodic Transmission:

- By periodically transmitting the current state of the link when no error conditions are present, the NOP.Debug Flits can serve as valuable checkpoints when debugging a failure and trying to identify the failure point with respect to regular link traffic.

2. Precursor to Error Conditions:

- The transmission of NOP.Debug Flits may be triggered as a preemptive signal when the system identifies patterns that typically precede error conditions, alerting an observer to the state of the link and providing context for debugging.

3. Response to Error Conditions:

- In the event of an error, NOP.Debug Flits can be transmitted immediately to capture the state of the link at the time of the error, providing valuable context for debugging.

4. Manual Trigger:

- Software-initiated transmission of NOP.Debug Flits allows for on-demand generation of debug information, giving the ability to manually trigger NOP.Debug Flits and control the number of Flits and debug content for targeted diagnostic purposes.

The transmission of NOP.Debug Flits operates on a best-effort basis, with no guarantee of delivery. The receiver has no requirements on how to handle the received Flits, but receiving NOP.Debug Flits must not affect the link state. This approach is designed to minimize the impact on the primary function of the PCI Express Link while still providing a channel for essential debug information. NOP.Debug and NOP.Vendor Flits are not replayed and are thus subject to loss due to bit errors.

In instances where direct physical access to a PCI Express Link is not possible (e.g., on-chip Links or Links where the traces are not wired to a connector might not be visible by a Protocol Analyzer), Switches and Root Complexes are permitted to forward NOP.Debug and/or NOP.Vendor Flits between Ports. This feature allows NOP.Debug and/or NOP.Vendor Flits to be routed to a different, accessible Link, ensuring that debug information remains observable regardless of the physical constraints of the system. Similarly, a single component may have multiple independent entities generating NOP.Debug or NOP.Vendor Flits that are then forwarded to a transmitting Port. In both situations, this forwarding occurs on a best effort basis and may be subject to Flit dropping on buffer overflows. The routing mechanism between Links is implementation specific, but each NOP.Debug and NOP.Vendor Flit includes a NOP Stream ID within the TLP bytes to identify its source. The forwarded NOP.Debug and NOP.Vendor Flits must preserve all TLP bytes of the original NOP.Debug Flit with the exception that the forwarding agent is permitted to overwrite the **NOP Flit Counter** to FFFh.

Forwarding between Ports of a Switch or Root Complex is not required. When implemented, a subset of Ports may be supported. For example, it is unlikely to make sense to support forwarding between Ports on different sockets of a multi-socket Root Complex. Similarly, to avoid implementing a parallel switching fabric, forwarding might only be supported between “adjacent” Ports of a Switch or Root Complex. The Debug Flit Maximum Rate Hint field can be used to help avoid buffer overflow due to aggregation at forwarding entities.

NOP.Debug and NOP.Vendor Flits carry a unique NOP Flit Counter for each NOP Stream ID. This counter allows observers to detect most missing or dropped Flits, providing an observer a reasonably reliable method for tracking the continuity of the debug information stream. Note that the NOP Flit Counter is a 12-bit field and cannot detect certain loss patterns.

Forwarding agents should change the NOP Flit Counter to FFFh to provide a hint to observers that a large number of Flits were dropped.

NOP.Debug Flits carry a Vendor ID, enabling the transmission of custom debug information tailored to specific vendor requirements. Additionally, PCI-SIG defined Debug Opcodes are available (see § [Table 4-25](#)), offering standardized options for debug operations and enhancing the interoperability of the mechanism across different vendor implementations.

Transmission of NOP.Debug and NOP.Vendor Flits must not interfere with entry and exit conditions for power management substates. Upon transition to a new state, transmission of NOP.Debug and NOP.Vendor Flits must be appropriately suspended or modified to align with the activity level permitted by the target state.

7. Software Initialization and Configuration §

The PCI Express Configuration model supports two Configuration Space access mechanisms:

- PCI-compatible Configuration Access Mechanism (CAM) (see § [Section 7.2.1](#))
- PCI Express Enhanced Configuration Access Mechanism (ECAM) (see § [Section 7.2.2](#))

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

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

7.1 Configuration Topology §

To maintain compatibility with PCI software configuration mechanisms, all PCI Express elements have a PCI-compatible Configuration Space. Each PCI Express Link originates from a logical PCI-PCI Bridge and is mapped into Configuration Space as the secondary bus of this Bridge. The Root Port is a PCI-PCI Bridge structure that originates a PCI Express Link from a PCI Express Root Complex (see § [Figure 7-1](#)).

A PCI Express Switch not using FPB Routing ID mechanisms is represented by multiple PCI-PCI Bridge structures connecting PCI Express Links to an internal logical PCI bus (see § [Figure 7-2](#)). The Switch Upstream Port is a PCI-PCI Bridge; the secondary bus of this Bridge represents the Switch's internal routing logic. Switch Downstream Ports are PCI-PCI Bridges bridging from the internal bus to buses representing the Downstream PCI Express Links from a PCI Express Switch. Only the PCI-PCI Bridges representing the Switch Downstream Ports may appear on the internal bus. Endpoints, represented by Type 0 Configuration Space Headers , are not permitted to appear on the internal bus.

A PCI Express Endpoint is mapped into Configuration Space as a single Function in a Device, which might contain multiple Functions or just that Function. PCI Express Endpoints and Legacy Endpoints are required to appear within one of the Hierarchy Domains originated by the Root Complex, meaning that they appear in Configuration Space in a tree that has a Root Port as its head. Root Complex Integrated Endpoints (RCiEPs) and Root Complex Event Collectors do not appear within one of the Hierarchy Domains originated by the Root Complex. These appear in Configuration Space as peers of the Root Ports.

Unless otherwise specified, requirements in the Configuration Space definition for a device apply to Single-Function Devices as well as to each Function individually of a Multi-Function Device .

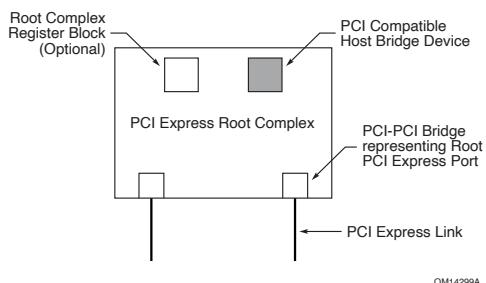


Figure 7-1 PCI Express Root Complex Device Mapping §

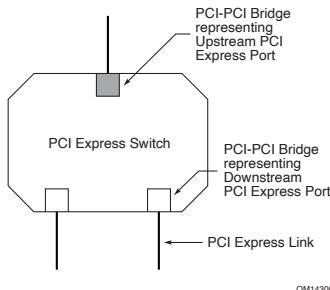


Figure 7-2 PCI Express Switch Device Mapping ¹⁶³ §

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: CHANGING A DEVICE'S OS DRIVER BINDINGS & RESOURCES §

Under certain circumstances, it is highly useful for a device to change one or more of its architected registers that help determine which OS-managed resources are allocated to the device and which OS drivers get bound to it. Here are two key examples:

- A device might change the Class Code Register value in one or more of its Functions
- Entire Functions might be added or removed after device firmware is updated

Software can direct devices to change their architected registers through a variety of mechanisms outside the scope of this specification, including implementation specific ones and ones defined by external standards bodies. It is recommended that these mechanisms follow the guidance of this Implementation Note.

If a device outside the RC has possibly been enumerated by the OS, it is strongly recommended that software use a mechanism directing the device to change its registers when coming out of a Conventional Reset. From a hardware perspective, this is similar to a hot remove followed by a hot add, providing a clean trigger point for architected registers to change in compliance with this specification, plus it guarantees the architected default hardware state for OS I/O infrastructure software and OS drivers to rely on. It is strongly recommended that software use OS-specific interfaces to perform the Conventional Reset and/or coordinate the reset and subsequent enumeration with the OS. If not feasible via OS-specific interfaces, software may be able to perform a Conventional Reset directly via several ways, including the Secondary Bus Reset bit or the Link Disable bit in the Downstream Port above the device.

If software knows that a device outside the RC has not been enumerated by the OS, software may choose to direct the device to change its registers without undergoing a reset, thus avoiding unnecessary delay.

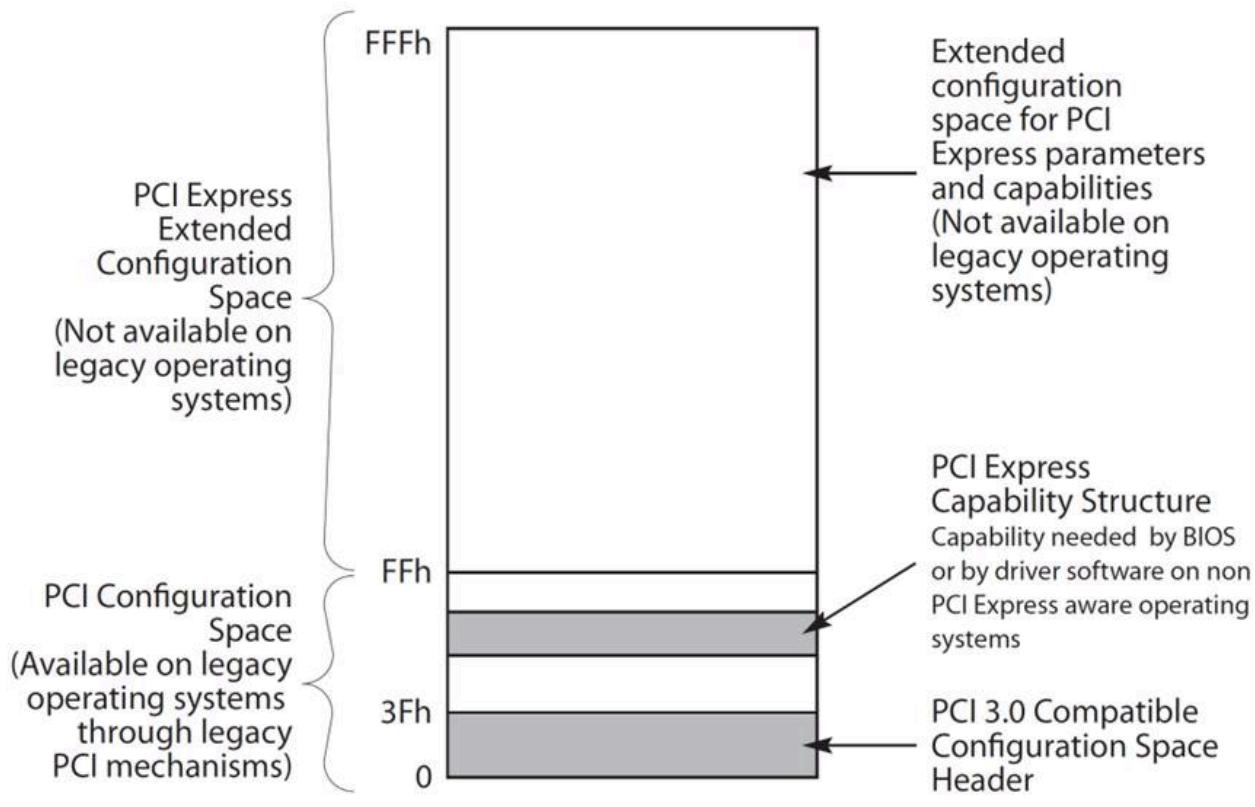
7.2 PCI Express Configuration Mechanisms §

PCI Express extends the Configuration Space to 4096 bytes per Function as compared to 256 bytes allowed by [PCI]. PCI Express Configuration Space is divided into a PCI-compatible region, which consists of the first 256 bytes of a Function's Configuration Space, and a PCI Express Extended Configuration Space which consists of the remaining Configuration Space (see § Figure 7-3). The PCI-compatible Configuration Space can be accessed using either the mechanism defined

¹⁶³. 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.

in § Section 7.2.1 or § Section 7.2.2 . Accesses made using either access mechanism are equivalent. The PCI Express Extended Configuration Space can only be accessed by using the ECAM mechanism defined in § Section 7.2.2 .¹⁶⁴

Base 6.4 vs Base 6.3



OM14301A

Figure 7-3 PCI Express Configuration Space Layout §

7.2.1 PCI-compatible Configuration Mechanism §

The PCI-compatible PCI Express Configuration Mechanism supports the PCI Configuration Space programming model defined in the [PCI]. By adhering to this model, systems incorporating PCI Express interfaces remain compliant with conventional PCI bus enumeration and configuration software.

In the same manner as PCI device Functions, PCI Express device Functions are required to provide a Configuration Space for software-driven initialization and configuration. Except for the differences described in this chapter, the PCI Express Configuration Space headers are organized to correspond with the format and behavior defined in the [PCI] (Section 6.1).

The PCI-compatible Configuration Access Mechanism uses the same Request format as the ECAM . For PCI-compatible Configuration Requests, the Extended Register Address field must be all zeros.

¹⁶⁴. The mechanism defined in § Section 7.2.1 and the ECAM mechanism defined in § Section 7.2.2 and operate independently from each other; there is no implied ordering between the two.

7.2.2 PCI Express Enhanced Configuration Access Mechanism (ECAM) §

For systems that are PC-compatible, or that do not implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the ***Enhanced Configuration Access Mechanism (ECAM)*** is required as defined in this section.

For systems that implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the operating system uses the standard firmware interface, and the hardware access mechanism defined in this section is not required.

In all systems, device drivers are encouraged to use the application programming interface (API) provided by the operating system to access the Configuration Space of its device and not directly use the hardware mechanism.

The ECAM utilizes a flat memory-mapped address space to access device configuration registers. In this case, the memory address determines the configuration register accessed and the memory data updates (for a write) or returns the contents of (for a read) the addressed register. The mapping from memory address space to PCI Express Configuration Space address is defined in § Table 7-1.

The size and base address for the range of memory addresses mapped to the Configuration Space are determined by the design of the host bridge and the firmware. They are reported by the firmware to the operating system in an implementation specific manner. The size of the range is determined by the number of bits that the host bridge maps to the Bus Number field in the configuration address. In § Table 7-1, this number of bits is represented as n , where $1 \leq n \leq 8$. A host bridge that maps n memory address bits to the Bus Number field supports Bus Numbers from 0 to $2^n - 1$, inclusive, and the base address of the range is aligned to a $2^{(n+20)}$ -byte memory address boundary. Any bits in the Bus Number field that are not mapped from memory address bits must be Clear.

For example, if a system maps three memory address bits to the Bus Number field, the following are all true:

- $n = 3$.
- Address bits A[63:23] are used for the base address, which is aligned to a 2^{23} -byte (8 MB) boundary.
- Address bits A[22:20] are mapped to bits [2:0] in the Bus Number field.
- Bits [7:3] in the Bus Number field are set to Clear.
- The system is capable of addressing Bus Numbers between 0 and 7, inclusive.

A minimum of one memory address bit ($n = 1$) must be mapped to the Bus Number field. Systems are encouraged to map additional memory address bits to the Bus Number field as needed to support a larger number of buses. Systems that support more than 4 GB of memory addresses are encouraged to map eight bits of memory address ($n = 8$) to the Bus Number field. Note that in systems that include multiple host bridges with different ranges of Bus Numbers assigned to each host bridge, the highest Bus Number for the system is limited by the number of bits mapped by the host bridge to which the highest bus number is assigned. In such a system, the highest Bus Number assigned to a particular host bridge would be greater, in most cases, than the number of buses assigned to that host bridge. In other words, for each host bridge, the number of bits mapped to the Bus Number field, n , must be large enough that the highest Bus Number assigned to each particular bridge must be less than or equal to $2^n - 1$ for that bridge.

In some processor architectures, it is possible to generate memory accesses that cannot be expressed in a single Configuration Request, for example due to crossing a DW aligned boundary, or because a locked access is used. A Root Complex implementation is not required to support the translation to Configuration Requests of such accesses.

Table 7-1 Enhanced Configuration Address Mapping §

| Memory Address ¹⁶⁵ | PCI Express Configuration Space |
|-------------------------------|--|
| A[(20+ n -1):20] | Bus Number $1 \leq n \leq 8$ |
| A[19:15] | Device Number |
| A[14:12] | Function Number |
| A[11:8] | Extended Register Number |
| A[7:2] | Register Number |
| A[1:0] | Along with size of the access, used to generate Byte Enables |

Note: for Requests targeting Extended Functions in an ARI Device, A[19:12] represents the (8-bit) Function Number, which replaces the (5-bit) Device Number and (3-bit) Function Number fields above.

The system hardware must provide a method for the system software to guarantee that a write transaction using the ECAM is completed by the completer before system software execution continues.

165. This address refers to the byte-level address from a software point of view.

IMPLEMENTATION NOTE: ORDERING CONSIDERATIONS FOR THE ENHANCED CONFIGURATION ACCESS MECHANISM §

The ECAM converts memory transactions from the host CPU into Configuration Requests on the PCI Express fabric. This conversion potentially creates ordering problems for the software, because writes to memory addresses are typically posted transactions but writes to Configuration Space are not posted on the PCI Express fabric.

Generally, software does not know when a posted transaction is completed by the completer. In those cases in which the software must know that a posted transaction is completed by the completer, one technique commonly used by the software is to read the location that was just written. For systems that follow the PCI ordering rules throughout, the read transaction will not complete until the posted write is complete. However, since the PCI ordering rules allow non-posted write and read transactions to be reordered with respect to each other, the CPU must wait for a non-posted write to complete on the PCI Express fabric to be guaranteed that the transaction is completed by the completer.

As an example, software may wish to configure a device Function's Base Address register by writing to the device using the ECAM, and then read a location in the memory-mapped range described by this Base Address register. If the software were to issue the memory-mapped read before the ECAM write was completed, it would be possible for the memory-mapped read to be re-ordered and arrive at the device before the Configuration Write Request, thus causing unpredictable results.

To avoid this problem, processor and host bridge implementations must ensure that a method exists for the software to determine when the write using the ECAM is completed by the completer.

This method may simply be that the processor itself recognizes a memory range dedicated for mapping ECAM accesses as unique, and treats accesses to this range in the same manner that it would treat other accesses that generate non-posted writes on the PCI Express fabric, i.e., that the transaction is not posted from the processor's viewpoint. An alternative mechanism is for the host bridge (rather than the processor) to recognize the memory-mapped Configuration Space accesses and not to indicate to the processor that this write has been accepted until the non-posted Configuration Transaction has completed on the PCI Express fabric. A third alternative would be for the processor and host bridge to post the memory-mapped write to the ECAM and for the host bridge to provide a separate register that the software can read to determine when the Configuration Write Request has completed on the PCI Express fabric. Other alternatives are also possible.

IMPLEMENTATION NOTE: GENERATING CONFIGURATION REQUESTS §

Because Root Complex implementations are not required to support the generation of Configuration Requests from accesses that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such accesses when using the memory-mapped ECAM unless it is known that the Root Complex implementation being used will support the translation.

7.2.2.1 Host Bridge Requirements §

For those systems that implement the ECAM, the PCI Express Host Bridge is required to translate the memory-mapped PCI Express Configuration Space accesses from the host processor to PCI Express configuration transactions. The use of Host Bridge PCI class code is Reserved for backwards compatibility; Host Bridge Configuration Space is opaque to standard PCI Express software and may be implemented in an implementation specific manner that is compatible with PCI Host Bridge Type 0 Configuration Space. A PCI Express Host Bridge is not required to signal errors through a Root Complex Event Collector. This support is optional for PCI Express Host Bridges.

7.2.2.2 PCI Express Device Requirements §

Devices must support an additional 4 bits for decoding configuration register access, i.e., they must decode the Extended Register Address[3:0] field of the Configuration Request header.

IMPLEMENTATION NOTE: DEVICE-SPECIFIC REGISTERS IN CONFIGURATION SPACE §

It is strongly recommended that PCI Express devices place no registers in Configuration Space other than those in headers or Capability structures architected by applicable PCI specifications.

Device-specific registers that have legitimate reasons to be placed in Configuration Space (e.g., they need to be accessible before Memory Space is allocated) should be placed in a Vendor-Specific Capability structure ([§ Section 7.9.4](#)), a Vendor-Specific Extended Capability structure ([§ Section 7.9.5](#), or [§ Section 7.9.6](#)).

Device-specific registers accessed in the run-time environment by drivers should be placed in Memory Space that is allocated by one or more Base Address registers. Even though PCI-compatible or PCI Express Extended Configuration Space may have adequate room for run-time device-specific registers, placing them there is highly discouraged for the following reasons:

- Not all Operating Systems permit drivers to access Configuration Space directly.
- Some platforms provide access to Configuration Space only via firmware calls, which typically have substantially lower performance compared to mechanisms for accessing Memory Space.
- Even on platforms that provide direct access to a memory-mapped PCI Express Enhanced Configuration Mechanism, performance for accessing Configuration Space will typically be significantly lower than for accessing Memory Space since:
 - Configuration Reads and Writes must usually be DWORD or smaller in size,
 - Configuration Writes are usually not posted by the platform, and
 - Some platforms support only one outstanding Configuration Write at a time.

IMPLEMENTATION NOTE: CONFIGURATION SPACE READ SIDE EFFECTS §

During a read access, any observable interaction that occurs besides the desired value being returned is called a read side effect. System software that has no specific knowledge of the Function being accessed may issue read requests to anywhere within the Function's Configuration Space. It is highly undesirable that any such access has any read side effects. No such side effects are required in any of the Configuration Space registers defined in this specification. It is strongly recommended that any implementation of those registers, as well as any vendor-defined Configuration Space registers, be free of any read side effects.

7.2.3 Root Complex Register Block (RCRB) §

A Root Port or RCiEP may be associated with an optional 4096-byte block of memory mapped registers referred to as the Root Complex Register Block (RCRB). These registers are used in a manner similar to Configuration Space and can include PCI Express Extended Capabilities and other implementation specific registers that apply to the Root Complex. The structure of the RCRB is described in § [Section 7.6.2](#).

Multiple Root Ports or internal devices are permitted to be associated with the same RCRB. The RCRB memory-mapped registers must not reside in the same address space as the memory-mapped Configuration Space or Memory Space.

A Root Complex implementation is not required to support accesses to an RCRB that cross DWORD aligned boundaries or accesses that use locked semantics.

IMPLEMENTATION NOTE: ACCESSING ROOT COMPLEX REGISTER BLOCK §

Because Root Complex implementations are not required to support accesses to a RCRB that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such accesses when accessing a RCRB unless the Root Complex will support the access.

7.3 Configuration Transaction Rules §

7.3.1 Device Number §

With non-ARI Devices, PCI Express components are restricted to implementing a single Device Number on their primary interface (Upstream Port), but are permitted to implement up to eight independent Functions within that Device Number. Each internal Function is selected based on decoded address information that is provided as part of the address portion of Configuration Request packets.

Except when FPB Routing ID mechanisms are used (see § [Section 6.26](#)), Downstream Ports that do not have ARI Forwarding enabled must associate only Device 0 with the device attached to the Logical Bus representing the Link from the Port. Configuration Requests targeting the Bus Number associated with a Link specifying Device Number 0 are delivered to the device attached to the Link; Configuration Requests specifying all other Device Numbers (1-31) must be terminated by the Switch Downstream Port or the Root Port with an Unsupported Request Completion Status (equivalent to Master Abort in PCI).

Non-ARI Devices must capture their assigned Device Number as discussed in § Section 2.2.6.2 . Non-ARI Devices must respond to all Type 0 Configuration Read Requests, regardless of the Device Number specified in the Request.

Switches, and components wishing to incorporate more than eight Functions at their Upstream Port, are permitted to implement one or more “virtual switches” represented by multiple Type 1 Configuration Space Headers (PCI-PCI Bridge) as illustrated in § Figure 7-2 . These virtual switches serve to allow fan-out beyond eight Functions. FPB provides a “flattening” mechanism that, when enabled, causes the virtual bridges of the Downstream Ports to appear in configuration space at RID addresses following the RID of the Upstream Port (see § Section 6.26). Since Switch Downstream Ports are permitted to appear on any Device Number, in this case all address information fields (Bus, Device, and Function Numbers) must be completely decoded to access the correct register. Any Configuration Request targeting an unimplemented Bus, Device, or Function must return a Completion with Unsupported Request Completion Status.

With an ARI Device, its Device Number is implied to be 0 rather than specified by a field within an ID. The traditional 5-bit Device Number and 3-bit Function Number fields in its associated Routing IDs, Requester IDs, and Completer IDs are interpreted as a single 8-bit Function Number. See § Section 6.13 . Any Type 0 Configuration Request targeting an unimplemented Function in an ARI Device must be handled as an Unsupported Request.

If an ARI Downstream Port has ARI Forwarding enabled, the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.

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

7.3.2 Configuration Transaction Addressing §

PCI Express Configuration Requests use the following addressing fields:

- Destination Segment (Flit Mode only) - Selects one of multiple Segments that may be implemented within a Root Complex. See § Section 2.2.1.2 for a list of Segment rules.
- Bus Number - PCI Express maps logical PCI Bus Numbers onto PCI Express Links such that PCI-compatible configuration software views the Configuration Space of a PCI Express Hierarchy as a PCI hierarchy including multiple bus segments.
- Device Number - Device Number association is discussed in § Section 7.3.1 and in § Section 6.26 . When an ARI Device is targeted and the Downstream Port immediately above it is enabled for ARI Forwarding, the Device Number is implied to be 0, and the traditional Device Number field is used instead as part of an 8-bit Function Number field. See § Section 6.13 .
- Function Number - PCI Express also supports Multi-Function Devices using the same discovery mechanism as PCI. A Multi-Function Device must fully decode the Function Number field. A Single-Function Device *MUST@FLIT* also fully decode the Function Number field. With ARI Devices, discovery and enumeration of Extended Functions require ARI-aware software. See § Section 6.13 .
- Extended Register Number and Register Number - Specify the Configuration Space address of the register being accessed (concatenated such that the Extended Register Number forms the more significant bits).

7.3.3 Configuration Request Routing Rules §

For Endpoints, the following rules apply:

- If Configuration Request Type is 1,
 - and the TLP is not an IDE TLP

- and it is targeting a Device’s captured Bus Number (See § [Section 9.2.1.2](#) for SRIOV devices that consume multiple Bus numbers)
 - Follow the rules for handling Unsupported Requests
- If Configuration Request Type is 0, or if Configuration Request Type is 1 and the TLP is an IDE TLP associated with a Selective IDE Stream in the Secure state,
 - Determine if the Request addresses a valid local Configuration Space of an implemented Function
 - If so, process the Request
 - If not, follow rules for handling Unsupported Requests

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

- Propagation of Configuration Requests from Downstream to Upstream as well as peer-to-peer are not supported
 - Configuration Requests are initiated only by the Host Bridge, including those passed through the [SFI CAM](#) mechanism
- If Configuration Request Type is 0, or if Configuration Request Type is 1 and the TLP is an IDE TLP associated with a Selective IDE Stream in the Secure state,
 - Determine if the Request addresses a valid local Configuration Space of an implemented Function
 - If so, process the Request
 - If not, follow rules for handling Unsupported Requests
- If Configuration Request Type is 1, apply the following tests, in sequence, to the Bus Number and Device Number fields:
 - If in the case of a PCI Express-PCI Bridge, equal to the Bus Number assigned to secondary PCI bus or, in the case of a Switch or Root Complex, equal to the Bus Number and decoded Device Numbers assigned to one of the Root (Root Complex) or Downstream Ports (Switch), or if required based on the FPB Routing ID mechanism,
 - Transform the Request to Type 0 by changing the value in the Type[4:0] field of the Request (see [Table 2-3](#)) - all other fields of the Request remain unchanged
 - Forward the Request to that Downstream Port (or PCI bus, in the case of a PCI Express-PCI Bridge)
 - If not equal to the Bus Number of any of Downstream Ports or secondary PCI bus, but in the range of Bus Numbers assigned to either a Downstream Port or a secondary PCI bus, or if required based on the FPB Routing ID mechanism,
 - Forward the Request to that Downstream Port interface without modification
 - Else (none of the above)
 - The Request is invalid - follow the rules for handling Unsupported Requests
- PCI Express-PCI Bridges must terminate as Unsupported Requests any Configuration Requests for which the Extended Register Address field is non-zero that are directed towards a PCI bus that does not support Extended Configuration Space.

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

Additional rule specific to Root Complexes:

- Configuration Requests addressing a Destination Segment and Bus Numbers assigned to devices within the Root Complex are processed by the Root Complex
 - The assignment of Segment Numbers and Bus Numbers to the devices within a Root Complex may be done in an implementation specific way.

For all types of devices:

Configuration Reads and Writes to unimplemented registers are not considered to be errors. Unless errors defined elsewhere in this specification are detected and need to be reported, such Requests must return a Completion with Successful Completion status, with reads returning a data value of all 0's and writes discarding the write data without effect.

All other Configuration Space addressing fields are decoded as described elsewhere in this specification.

7.3.4 PCI Special Cycles §

PCI Special Cycles (see the [PCI] for details) are not directly supported by PCI Express. PCI Special Cycles may be directed to PCI bus segments behind PCI Express-PCI Bridges using Type 1 configuration cycles as described in the [PCI].

7.4 Configuration Register Types §

Configuration register fields are assigned one of the attributes described in § Table 7-2 . All PCI Express components, with the exception of the Root Complex and system-integrated devices, initialize register fields to specified default values. Root Complexes and system-integrated devices initialize register fields as required by the firmware for a particular system implementation.

Table 7-2 Register and Register Bit-Field Types §

| Register Attribute | Description |
|--------------------|---|
| HwInit | Hardware Initialized - Register bits are permitted, as an implementation option, to be hard-coded, initialized by system/device firmware, or initialized by hardware mechanisms such as pin strapping or nonvolatile storage. ¹⁶⁶ Initialization by system firmware is permitted only for system-integrated devices. Bits must be fixed in value and read-only after initialization. After Initialization, values are only permitted to change following Conventional Reset (see § Section 6.6.1) and subsequent re-initialization. HwInit register bits are not modified by an FLR . |
| RO | Read-only - Register bits are read-only and cannot be altered by software. Where explicitly defined, these bits are used to reflect changing hardware state, and as a result bit values can be observed to change at run time. ¹⁶⁷ Register bit default values and bits that cannot change value at run time, are permitted to be hard-coded, initialized by system/ device firmware, or initialized by hardware mechanisms such as pin strapping or nonvolatile storage. Initialization by system firmware is permitted only for system-integrated devices. If the optional feature that would Set the bits is not implemented, the bits must be hardwired to Zero. |
| RW | Read-Write - Register bits are read-write and are permitted to be either Set or Cleared by software to the desired state. If the optional feature that is associated with the bits is not implemented, the bits are permitted to be hardwired to Zero. |
| RW1C | Write-1-to-clear status - Register bits indicate status when read. A Set bit indicates a status event which is Cleared by writing a 1b. Writing a 0b to RW1C bits has no effect. If the optional feature that would Set the bit is not implemented, the bit must be read-only and hardwired to Zero. |

166. For historical reasons, readers may observe inconsistencies in this document in the use of HwInit and RO . As this document is revised we will attempt to ensure that new definitions conform to the definitions given here.

167. For historical reasons, readers may observe inconsistencies in this document in the use of HwInit and RO . As this document is revised we will attempt to ensure that new definitions conform to the definitions given here.

Base 6.4 vs Base 6.3

| Register Attribute | Description |
|--------------------|--|
| ROS | <p>Sticky - Read-only - Register bits are read-only and cannot be altered by software. If the optional feature that would Set the bit is not implemented, the bit is hardwired to Zero. Bits are neither initialized nor modified by hot reset or FLR.¹⁶⁸</p> <p>Where noted, devices that consume auxiliary power must preserve sticky register bit values when auxiliary power consumption (via either Aux Power PM Enable or PME_En) is enabled. In these cases, register bits are neither initialized nor modified by Hot, Warm, or Cold Reset (see § Section 6.6).</p> |
| RWS | <p>Sticky - Read-Write - Register bits are read-write and are Set or Cleared by software to the desired state. Bits are neither initialized nor modified by hot reset or FLR.¹⁶⁹</p> <p>If the optional feature that is associated with the bits is not implemented, the bits are permitted to be hardwired to Zero.</p> <p>Where noted, devices that consume auxiliary power must preserve sticky register bit values when auxiliary power consumption (via either Aux Power PM Enable or PME_En) is enabled. In these cases, register bits are neither initialized nor modified by Hot, Warm, or Cold Reset (see § Section 6.6).</p> |
| RW1CS | <p>Sticky - Write-1-to-clear status - Register bits indicate status when read. A Set bit indicates a status event which is Cleared by writing a 1b. Writing a 0b to RW1CS bits has no effect. If the optional feature that would Set the bit is not implemented, the bit is read-only and hardwired to Zero. Bits are neither initialized nor modified by hot reset or FLR.¹⁷⁰</p> <p>Where noted, devices that consume auxiliary power must preserve sticky register bit values when auxiliary power consumption (via either Aux Power PM Enable or PME_En) is enabled. In these cases, register bits are neither initialized nor modified by Hot, Warm, or Cold Reset (see § Section 6.6).</p> |
| RsvdP | Reserved and Preserved - Reserved for future RW implementations. Register bits are read-only and must return zero when read. Software must preserve the value read for writes to bits. |
| RsvdZ | Reserved and Zero - Reserved for future RW1C implementations. Register bits are read-only and must return zero when read. Software must use 0b for writes to bits. |

For ~~↑↓SR-IOV devices, ↑↓SR-IOV Devices, ↑~~ many registers or fields in VFs are required to be reserved or hardwired to Zero. Before the Single Root I/O Virtualization and Sharing (SR-IOV) Specification was merged into the PCI Express Base Specification, the SR-IOV specification contained many tables summarizing requirements differences for PFs and VFs, relative to the Base specification. These tables contained dedicated columns for PF attributes and VF attributes, though there were relatively few differences for PF attributes.

To provide a clear, consolidated, and concise indication of PF and VF attribute differences from other Function types, this specification eliminated most of those tables. Instead, special field types from the following table indicate VF attribute differences, and any additional attribute or semantic differences with PFs and VFs are covered within the register description column or elsewhere.

168. Bits/fields with the “Sticky” attribute must be implemented such that no Function-specific software or firmware is required to maintain the observed state of the bit/field. Particularly for power management scenarios, it is permitted, but not recommended, to use Function-specific software or firmware to restore the correct values, provided this is done before the system hardware or system software could observe incorrect values. How this could be done is outside the scope of this document.

169. Bits/fields with the “Sticky” attribute must be implemented such that no Function-specific software or firmware is required to maintain the observed state of the bit/field. Particularly for power management scenarios, it is permitted, but not recommended, to use Function-specific software or firmware to restore the correct values, provided this is done before the system hardware or system software could observe incorrect values. How this could be done is outside the scope of this document.

170. Bits/fields with the “Sticky” attribute must be implemented such that no Function-specific software or firmware is required to maintain the observed state of the bit/field. Particularly for power management scenarios, it is permitted, but not recommended, to use Function-specific software or firmware to restore the correct values, provided this is done before the system hardware or system software could observe incorrect values. How this could be done is outside the scope of this document.

Table 7-3 Special Field Types to Indicate VF Attributes §

| Register Attribute | Description |
|--------------------|---|
| VF ROZ | VF RO-Zero - VF register bits must have RO semantics and be hardwired to Zero. |
| VF RsvdP | VF RsvdP - VF register bits must have RsvdP semantics. |
| VF RsvdZ | VF RsvdZ - VF register bits must have RsvdZ semantics. |

7.5 PCI and PCIe Capabilities Required by the Base Spec for all Ports §

The following registers and capabilities are required by this specification in all Functions, including PFs and VFs.

Except where noted, VF register fields and bits have the same attributes as other Functions. For VF fields marked RsvdP, the associated PF's setting applies to the VF.

7.5.1 PCI-Compatible Configuration Registers §

The first 256 bytes of a Function's Configuration Space form the PCI-compatible region. This region completely aliases the conventional PCI Configuration Space of the Function. Legacy PCI devices can also be accessed with the ECAM without requiring any modifications to the device hardware or device driver software.

Layout of the Configuration Space and format of individual configuration registers are depicted following the little-endian convention.

7.5.1.1 Type 0/1 Common Configuration Space §

§ Figure 7-4 details allocation for common register fields of Type 0 and Type 1 Configuration Space Headers for PCI Express Device Functions. Fields labeled Type Specific vary between different Configuration Space header types.

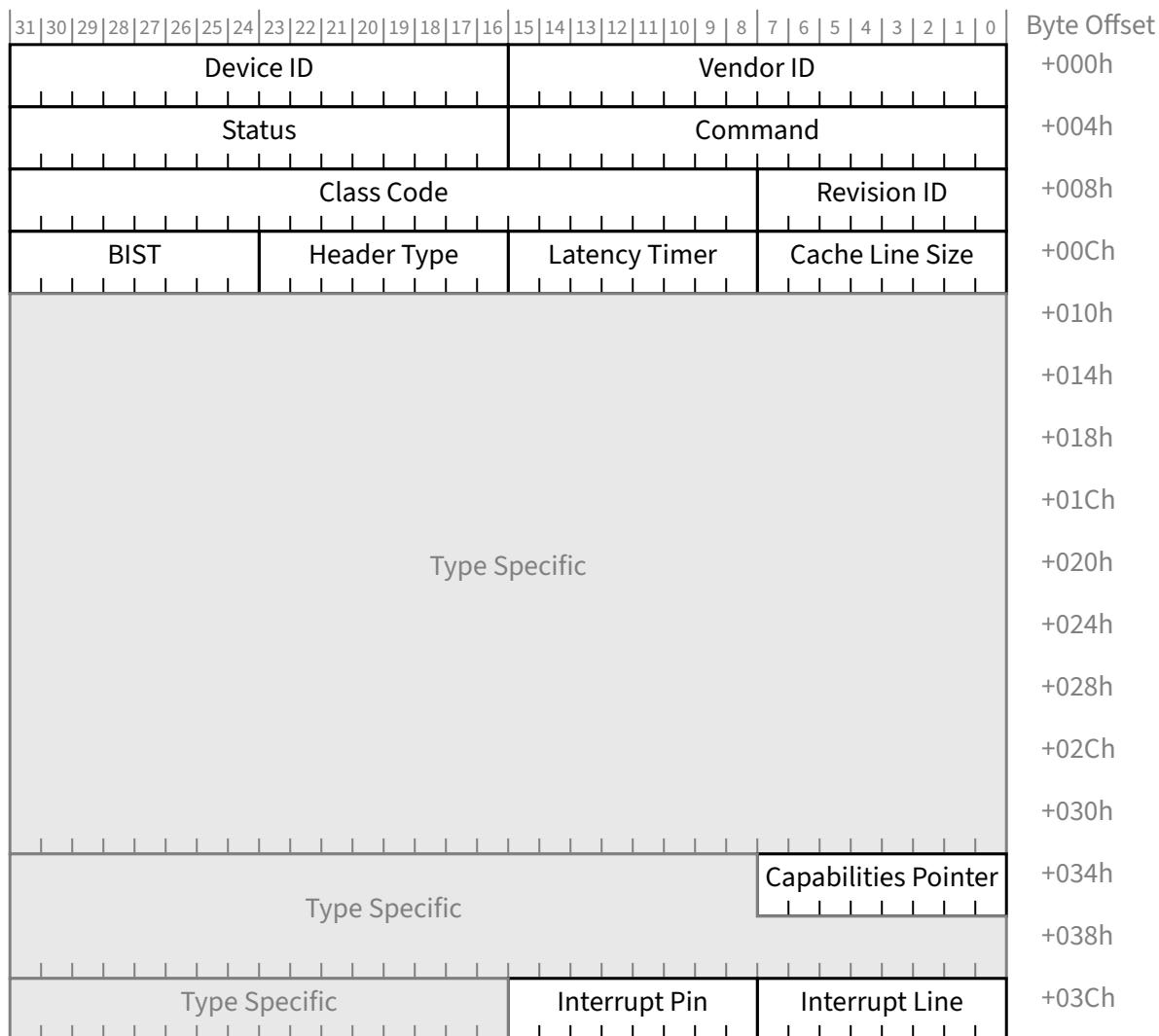


Figure 7-4 Common Configuration Space Header

These registers are defined for both Type 0 and Type 1 Configuration Space Headers. The PCI Express-specific interpretation of these registers is defined in this section.

7.5.1.1.1 Vendor ID Register (Offset 00h)

For non-VFs, the Vendor ID register is HwInit and the value in this register identifies the manufacturer of the Function. In keeping with PCI-SIG procedures, valid vendor identifiers must be allocated by the PCI-SIG to ensure uniqueness. Each vendor must have at least one Vendor ID. It is recommended that software read the Vendor ID register to determine if a Function is present, where a value of FFFFh indicates that no Function is present.

For VFs, this field must return FFFFh when read. VI software should return the Vendor ID value from the associated PF as the Vendor ID value for the VF.

7.5.1.1.2 Device ID Register (Offset 02h) §

For non-VFs, the Device ID register is HwInit and the value in this register identifies the particular Function. The Device ID must be allocated by the vendor. The Device ID, in conjunction with the Vendor ID and Revision ID, are used as one mechanism for software to determine which driver should be loaded. The vendor must ensure that the chosen values do not result in the use of an incompatible device driver.

For VFs, this field must return FFFFh when read. VI software should return the VF Device ID (see § [Section 9.4.3.11](#)) value from the associated PF as the Device ID for the VF.

IMPLEMENTATION NOTE: LEGACY PCI PROBING SOFTWARE §

Returning FFFFh for Device ID and Vendor ID values allows some legacy software to ignore VFs. See § [Section 7.5.1.1.1](#).

7.5.1.1.3 Command Register (Offset 04h) §

§ Table 7-4 defines the Command Register and the layout of the register is shown in § [Figure 7-5](#). Individual bits in the Command Register may or may not be implemented depending on the feature set supported by the Function. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register.

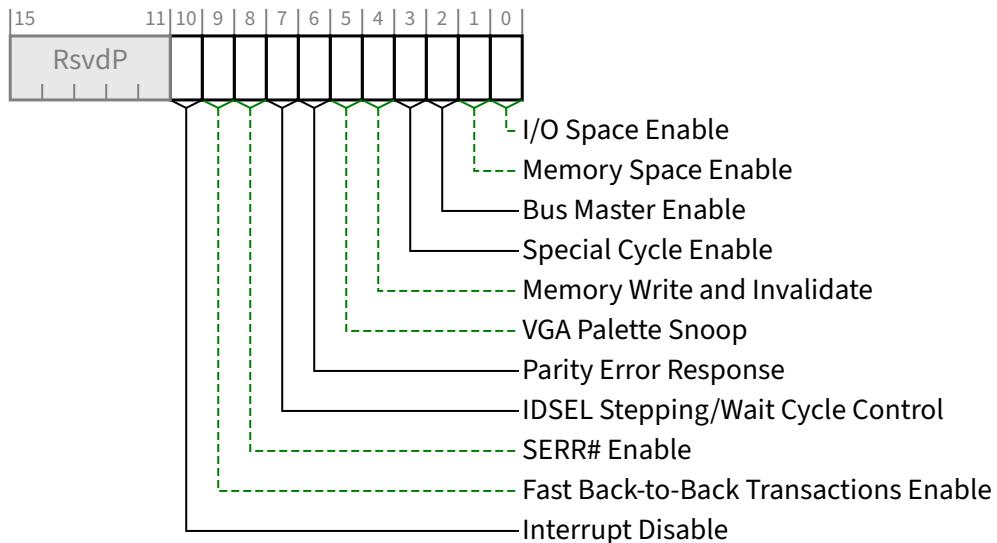


Figure 7-5 Command Register §

Base 6.4 vs Base 6.3

Table 7-4 Command Register §

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| 0 | <p>I/O Space Enable - Controls a Function's response to I/O Space accesses. When this bit is Clear, all received I/O accesses are caused to be handled as Unsupported Requests. When this bit is Set, the Function is enabled to decode the address and further process I/O Space accesses. For a Function with a Type 1 Configuration Space Header, this bit controls the response to I/O Space accesses received on its Primary Side.</p> <p>Default value of this bit is 0b.</p> <p>This bit is permitted to be hardwired to Zero if a Function does not support I/O Space accesses.</p> <p>This bit does not apply to VFs and must be hardwired to Zero.</p> | RW VF ROZ |
| 1 | <p>Memory Space Enable - Controls a Function's response to Memory Space accesses. When this bit is Clear, all received Memory Space accesses are caused to be handled as Unsupported Requests. When this bit is Set, the Function is enabled to decode the address and further process Memory Space accesses. For a Function with a Type 1 Configuration Space Header, this bit controls the response to Memory Space accesses received on its Primary Side.</p> <p>Default value of this bit is 0b.</p> <p>This bit is permitted to be hardwired to 0b if a Function does not support Memory Space accesses.</p> <p>This bit does not apply to VFs and must be hardwired to Zero. VF Memory Space is controlled by the VF MSE bit in the SR-IOV Control Register.</p> | RW VF ROZ |
| 2 | <p>Bus Master Enable - Controls the ability of a Function to issue Memory¹⁷¹ and I/O Read/Write Requests, and the ability of a Port to forward Memory and I/O Read/Write Requests in the Upstream direction</p> <ul style="list-style-type: none"> Functions with a Type 0 Configuration Space Header : <p>When this bit is Set, the Function is allowed to issue Memory or I/O Requests.</p> <p>When this bit is Clear, the Function is not allowed to issue any Memory or I/O Requests.</p> <p>Note that as MSI/MSI-X interrupt Messages are in-band memory writes, setting the Bus Master Enable bit to 0b disables MSI/MSI-X interrupt Messages as well.</p> <p>Transactions for a VF that has its Bus Master Enable Set must not be blocked by transactions for VFs that have their Bus Master Enable Cleared.</p> <p>Requests other than Memory or I/O Requests are not controlled by this bit.</p> <p>Default value of this bit is 0b.</p> <p>This bit is hardwired to 0b if a Function does not generate Memory or I/O Requests.</p> Functions with a Type 1 Configuration Space Header: <p>This bit controls the initiating of and the forwarding of Memory or I/O Requests by a Port in the Upstream direction. When this bit is 0b, Memory and I/O Requests received at a Root Port or the Downstream side of a Switch Port must be handled as Unsupported Requests (UR), and for Non-Posted Requests [↑ and UIO Requests] a Completion with UR Completion Status must be returned. This bit does not affect forwarding of Completions in either the Upstream or Downstream direction.</p> <p>The forwarding of Requests other than Memory or I/O Requests is not controlled by this bit.</p> <p>Default value of this bit is 0b.</p> | RW |
| 3 | <p>Special Cycle Enable - This bit was originally described in the [PCI]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.</p> | RO |

171. The AtomicOp Requester Enable bit in the Device Control 2 register must also be Set in order for an AtomicOp Requester to initiate AtomicOp Requests, which are Memory Requests.

 Errata: Base 6.3
B834△↔

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| 4 | Memory Write and Invalidate - This bit was originally described in the [PCI] and the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register. | RO |
| 5 | VGA Palette Snoop - This bit was originally described in the [PCI] and the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 6 | <p>Parity Error Response - See § Section 7.5.1.1.14 .</p> <p>This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Status Register . An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP |
| 7 | IDSEL Stepping/Wait Cycle Control - This bit was originally described in the [PCI]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 8 | <p>SERR# Enable - See § Section 7.5.1.1.14 .</p> <p>When Set, this bit enables reporting upstream of Non-fatal and Fatal errors detected by the Function. Note that errors are reported if enabled either through this bit or through the PCI Express specific bits in the Device Control Register (see § Section 7.5.3.4).</p> <p>In addition, for Functions with Type 1 Configuration Space Headers , this bit controls transmission by the primary interface of ERR_NONFATAL and ERR_FATAL error Messages forwarded from the secondary interface. This bit does not affect the transmission of forwarded ERR_COR messages.</p> <p>An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP |
| 9 | Fast Back-to-Back Transactions Enable - This bit was originally described in the [PCI]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 10 | <p>Interrupt Disable - Controls the ability of a Function to generate INTx emulation interrupts. When Set, Functions are prevented from asserting INTx interrupts.</p> <p>Any INTx emulation interrupts already asserted by the Function must be deasserted when this bit is Set. As described in § Section 2.2.8.1 , INTx interrupts use virtual wires that must, if asserted, be deasserted using the appropriate Deassert_INTx message(s) when this bit is Set.</p> <p>Only the INTx virtual wire interrupt(s) associated with the Function(s) for which this bit is Set are affected.</p> <p>For Functions with a Type 0 Configuration Space Header that generate INTx interrupts, this bit is required. For Functions with a Type 0 Configuration Space Header that do not generate INTx interrupts, this bit is optional. If not implemented, this bit must be hardwired to 0b.</p> <p>For Functions with a Type 1 Configuration Space Header that generate INTx interrupts on their own behalf, this bit is required. This bit has no effect on interrupts forwarded from the secondary side.</p> <p>For Functions with a Type 1 Configuration Space Header that do not generate INTx interrupts on their own behalf this bit is optional. If not implemented, this bit must be hardwired to 0b.</p> <p>Default value of this bit is 0b.</p> <p>This bit does not apply to VFs and must be hardwired to Zero.</p> | RW VF ROZ |

7.5.1.1.4 Status Register (Offset 06h) §

§ Table 7-5 defines the Status Register and the layout of the register is shown in § Figure 7-6 . Functions may not need to implement all bits, depending on the feature set supported by the Function. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register.

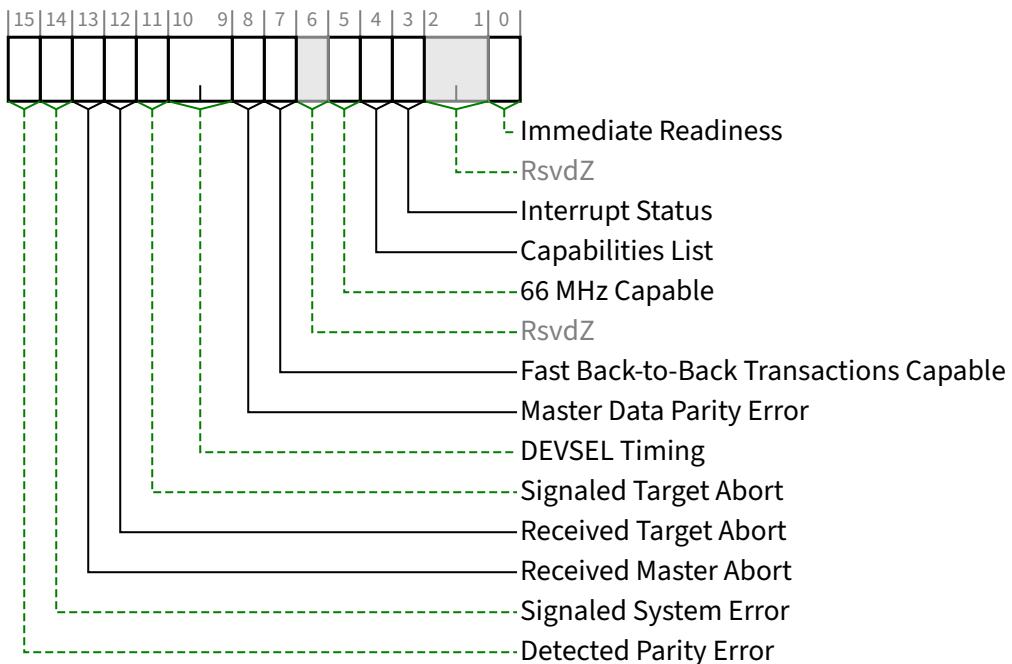


Figure 7-6 Status Register §

Table 7-5 Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| 0 | <p>Immediate Readiness - This optional bit, when Set, indicates the Function is guaranteed to be ready to successfully complete valid Configuration Requests at any time. It is permitted for this indication to be based on implementation specific knowledge of how long it takes the host to become ready to issue Configuration Requests.</p> <p>When this bit is Set, for accesses to this Function, software is exempt from all requirements to delay configuration accesses following any type of reset, including but not limited to the timing requirements defined in § Section 6.6 .</p> <p>How this guarantee is established is beyond the scope of this document.</p> <p>It is permitted that system software/firmware provide mechanisms that supersede the indication provided by this bit, however such software/firmware mechanisms are outside the scope of this specification.</p> <p>This bit does not apply to VFs and must be hardwired to Zero .</p> | RO VF ROZ |
| 3 | <p>Interrupt Status - When Set, indicates that an INTx emulation interrupt is pending internally in the Function.</p> | RO VF ROZ |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>Note that INTx emulation interrupts forwarded by Functions with a <u>Type 1 Configuration Space Header</u> from the secondary side are not reflected in this bit.</p> <p>Setting the Interrupt Disable bit has no effect on the state of this bit.</p> <p>Functions that do not generate INTx interrupts are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> <p>This bit does not apply to VFs and must be hardwired to Zero.</p> | |
| 4 | Capabilities List - Indicates the presence of an Extended Capability list item. Since all PCI Express device Functions are required to implement the PCI Express Capability structure, this bit must be hardwired to 1b. | RO |
| 5 | 66 MHz Capable - This bit was originally described in the [PCI]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 7 | Fast Back-to-Back Transactions Capable - This bit was originally described in the [PCI]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 8 | <p>Master Data Parity Error - See § Section 7.5.1.1.4</p> <p>This bit is Set by a Function with a Type 0 Configuration Space Header if the <u>Parity Error Response</u> bit in the <u>Command Register</u> is 1b and either of the following two conditions occurs:</p> <ul style="list-style-type: none"> • Function receives a Poisoned Completion • Function transmits a Poisoned Request <p>This bit is Set by a Function with a Type 1 Configuration Space Header if the <u>Parity Error Response</u> bit in the <u>Command Register</u> is 1b and either of the following two conditions occurs:</p> <ul style="list-style-type: none"> • Port receives a Poisoned Completion going Downstream • Port transmits a Poisoned Request Upstream <p>If the <u>Parity Error Response</u> bit is 0b, this bit is never Set.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 10:9 | DEVSEL Timing - This field was originally described in the [PCI]. Its functionality does not apply to PCI Express and the field must be hardwired to 00b. | RO |
| 11 | <p>Signaled Target Abort - See § Section 7.5.1.1.14.</p> <p>This bit is Set when a Function completes a <u>↑↓Posted↓↑↑Posted</u>, <u>Non-Posted,↑</u> or <u>↑↓Non-Posted↓↑↑UIO↑</u> Request as a Completer Abort error.</p> <p>This applies to a Function with a Type 1 Configuration Space Header when the Completer Abort was generated by its Primary Side.</p> <p>Functions with a Type 0 Configuration Space Header that do not signal Completer Abort are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 12 | <p>Received Target Abort - See § Section 7.5.1.1.14 .</p> <p>On a Function with a Type 0 Configuration Space Header, this bit is Set when a Requester receives a Completion with Completer Abort Completion Status.</p> <p>On a Function with a Type 1 Configuration Space Header , this bit is Set when its Primary Side receives a Completion with Completer Abort Completion Status.</p> | RW1C |

 Errata: Base 6.3
 B834△<ID>

| Bit Location | Register Description | Attributes |
|--------------|--|--|
| | <p>Functions with a Type 0 Configuration Space Header that do not make Non-Posted Requests #or UIO Requests on their own behalf are permitted to hardwire this bit to 0b. Default value of this bit is 0b.</p> | Errata: Base 6.3 B834△◀▶ |
| 13 | <p>Received Master Abort - See § Section 7.5.1.1.14.</p> <p>On a Function with a Type 0 Configuration Space Header , this bit is Set when a Requester receives a Completion with Unsupported Request Completion Status.</p> <p>On a Function with a Type 1 Configuration Space Header , the bit is Set when its Primary Side receives a Completion with Unsupported Request Completion Status.</p> <p>Functions with a Type 0 Configuration Space Header that do not make Non-Posted Requests #or UIO Requests on their own behalf are permitted to hardwire this bit to 0b. Default value of this bit is 0b.</p> | Errata: Base 6.3 B834△◀▶ |
| 14 | <p>Sigaled System Error - See § Section 7.5.1.1.14.</p> <p>This bit is Set when a Function sends an <u>ERR_FATAL</u> or <u>ERR_NONFATAL</u> Message, and the SERR# Enable bit in the <u>Command Register</u> is 1b.</p> <p>Functions with a Type 0 Configuration Space Header that do not send <u>ERR_FATAL</u> or <u>ERR_NONFATAL</u> Messages are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 15 | <p>Detected Parity Error - See § Section 7.5.1.1.14.</p> <p>This bit is Set by a Function whenever it receives a Poisoned TLP, regardless of the state of the Parity Error Response bit in the Command Register . On a Function with a Type 1 Configuration Space Header , the bit is Set when the Poisoned TLP is received by its Primary Side.</p> <p>Default value of this bit is 0b.</p> | RW1C |

7.5.1.1.5 Revision ID Register (Offset 08h) §

The Revision ID Register is HwInit and the value in this register specifies a Function specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. The Device ID, in conjunction with the Vendor ID and Revision ID, are used as one mechanism for software to determine which driver should be loaded. The vendor must ensure that the chosen values do not result in the use of an incompatible device driver.

The value reported in the VF may be different than the value reported in the PF.

7.5.1.1.6 Class Code Register (Offset 09h) §

The Class Code Register is read-only and is used to identify the generic operation of the Function and, in some cases, a specific register level programming interface. The register layout is shown in § [Figure 7-7](#) and described in § [Table 7-6](#). Encodings for base class, sub-class, and programming interface are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved.

The field in a PF and its associated VFs must return the same value when read.

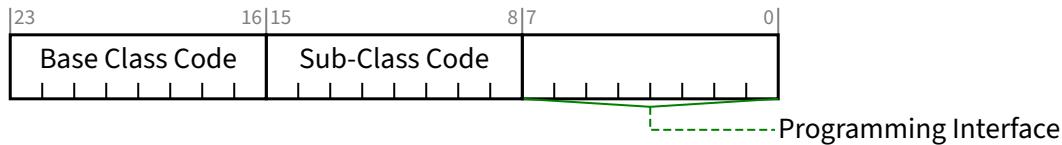


Figure 7-7 Class Code Register §

Table 7-6 Class Code Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Programming Interface - This field identifies a specific register-level programming interface (if any) so that device independent software can interact with the Function. Encodings for this field are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved. | RO |
| 15:8 | Sub-Class Code - Specifies a base class sub-class, which identifies more specifically the operation of the Function. Encodings for sub-class are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved. | RO |
| 23:16 | Base Class Code - A code that broadly classifies the type of operation the Function performs. Encodings for base class, are provided in the [PCI-Code-and-ID]. All unspecified encodings are Reserved. | RO |

7.5.1.1.7 Cache Line Size Register (Offset 0Ch) §

The Cache Line Size register is programmed by the system firmware or the operating system to system cache line size. However, note that legacy PCI-compatible software may not always be able to program this register correctly especially in the case of Hot-Plug devices. This read-write register is implemented for legacy compatibility purposes but has no effect on any PCI Express device behavior. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register. The default value of this register is 00h.

This bit does not apply to VFs and must be hardwired to Zero.

7.5.1.1.8 Latency Timer Register (Offset 0Dh) §

This register is also referred to as Primary Latency Timer for Type 1 Configuration Space Header Functions. The Latency Timer was originally described in the [PCI] and the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express. This register must be hardwired to 00h.

7.5.1.1.9 Header Type Register (Offset 0Eh) §

This register identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether or not the Device might contain multiple Functions. The register layout is shown in § Figure 7-8 and § Table 7-7 describes the bits in the register.

This entire register does not apply to VFs and must be hardwired to Zero.

Base 6.4 vs Base 6.3

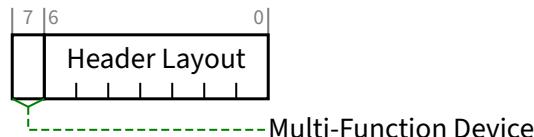


Figure 7-8 Header Type Register §

Table 7-7 Header Type Register §

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| 6:0 | <p>Header Layout - This field identifies the layout of the second part of the predefined header.</p> <p>For Functions that implement a Type 0 Configuration Space Header the encoding 000 0000b must be used.</p> <p>For Functions that implement a Type 1 Configuration Space Header the encoding 000 0001b must be used.</p> <p>The encoding 000 0010b is Reserved. This encoding was originally described in the [PC-Card] and is used in previous versions of the programming model. Careful consideration should be given to any attempt to repurpose it.</p> <p>All other encodings are Reserved.</p> | RO VF ROZ |
| 7 | <p>Multi-Function Device - When Set, indicates that the Device may contain multiple Functions, but not necessarily. Software is permitted to probe for Functions other than Function 0. When Clear, software must not probe for Functions other than Function 0 unless explicitly indicated by another mechanism, such as an ARI or SR-IOV Extended Capability structure. Except where stated otherwise, it is recommended that this bit be Set if there are multiple Functions, and Clear if there is only one Function.</p> <p>The presence of ↑↓Shadow Functions↓ ↑↑Shadow Functions↑ does not affect this bit.</p> <p>For an ↑↓SR-IOV device,↓ ↑↑SR-IOV Device,↑ this bit is Set in non-VFs only if there are multiple non-VFs. VFs do not affect the value of bit 7.</p> | RO VF ROZ |

7.5.1.1.10 BIST Register (Offset 0Fh) §

This register is used for control and status of BIST. Functions that do not support BIST must hardwire the register to 00h. VFs shall not support BIST. A Function whose BIST is invoked must not prevent normal operation of the PCI Express Link. § Table 7-8 describes the bits in the register and § Figure 7-9 shows the register layout.

For an **↑↓SR-IOV device,↓ ↑↑SR-IOV Device,↑** if the VF Enable bit is Set in any PF, then software should not invoke BIST in any Function associated with that device.

↑↑For an SIOV Device , if the SDI Enabled bit is Set in any PF, then software should not invoke BIST in any Function associated with that device.↑

ECN: Base 6.3
SIOV△↔

Base 6.4 vs Base 6.3

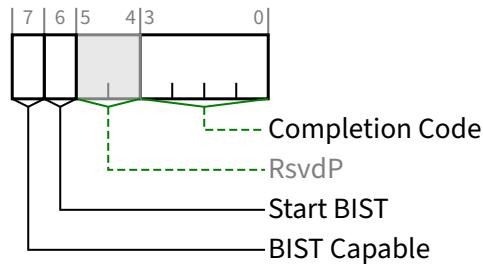


Figure 7-9 BIST Register §

Table 7-8 BIST Register §

| Bit Location | Register Description | Attributes |
|--------------|--|---|
| 3:0 | <p>Completion Code - This field encodes the status of the most recent test. A value of 0000b means that the Function has passed its test. Non-zero values mean the Function failed. Function-specific failure codes can be encoded in the non-zero values.</p> <p>This field's value is only meaningful when BIST Capable is Set and Start BIST is Clear.</p> <p>Default value of this field is 0000b.</p> <p>This field must be hardwired to 0000b if BIST Capable is Clear.</p> | <u>RO</u> <u>VF ROZ</u> |
| 6 | <p>Start BIST - If BIST Capable is Set, Set this bit to invoke BIST. The Function resets the bit when BIST is complete. Software is permitted to fail the device if this bit is not Clear (BIST is not complete) 2 seconds after it had been Set.</p> <p>Writing this bit to 0b has no effect.</p> <p>This bit must be hardwired to 0b if BIST Capable is Clear.</p> | <u>RW / RO</u> <u>(see description)</u> <u>VF ROZ</u> |
| 7 | BIST Capable - When Set, this bit indicates that the Function supports BIST. When Clear, the Function does not support BIST. | <u>HwInit</u> <u>VF ROZ</u> |

7.5.1.1.11 Capabilities Pointer (Offset 34h) §

This register is used to point to a linked list of capabilities implemented by this Function. Since all PCI Express Functions are required to implement the PCI Express Capability structure, which must be included somewhere in this linked list; this register must be non-zero. The bottom two bits are Reserved and must be set to 00b. Software must mask these bits off before using this register as a pointer in Configuration Space to the first entry of a linked list of new capabilities. This register is HwInit.

7.5.1.1.12 Interrupt Line Register (Offset 3Ch) §

The Interrupt Line register communicates interrupt line routing information. The register is read/write and must be implemented by any Function that uses an interrupt pin (see following description). Values in this register are programmed by system software and are system architecture specific. The Function itself does not use this value; rather the value in this register is used by device drivers and operating systems. If Interrupt Pin Register is 00h, this register is permitted to be hardwired to 0b. Otherwise, the default value is implementation specific.

For VFs, this register does not apply and must be hardwired to Zero.

7.5.1.1.13 Interrupt Pin Register (Offset 3Dh) §

The Interrupt Pin register is a read-only register that identifies the legacy interrupt Message(s) the Function uses (see § Section 6.1 for further details). Valid values are 01h, 02h, 03h, and 04h that map to legacy interrupt Messages for INTA, INTB, INTC, and INTD respectively. A value of 00h indicates that the Function uses no legacy interrupt Message(s). The values 05h through FFh are Reserved.

PCI Express defines one legacy interrupt Message for a Single-Function Device and up to four legacy interrupt Messages for a Multi-Function Device . For a Single-Function Device , only INTA may be used.

Any Function on a Multi-Function Device can use any of the INTx Messages. If a device implements a single legacy interrupt Message, it must be INTA; if it implements two legacy interrupt Messages, they must be INTA and INTB; and so forth. For a Multi-Function Device , all Functions may use the same INTx Message or each may have its own (up to a maximum of four Functions) or any combination thereof. A single Function can never generate an interrupt request on more than one INTx Message.

For VFs, this register does not apply and must be hardwired to Zero.

7.5.1.1.14 Error Registers §

The Error Control/Status register bits in the Command and Status registers (see § Section 7.5.1.1.3 and § Section 7.5.1.1.4 respectively) and the Bridge Control Register and Secondary Status Register of Type 1 Configuration Space Header Functions (see § Section 7.5.1.3.10 and § Section 7.5.1.3.7 respectively) control PCI-compatible error reporting for both PCI and PCI Express device Functions. Mapping of PCI Express errors onto PCI errors is also discussed in § Section 6.2.7.1 . In addition to the PCI-compatible error control and status, PCI Express error reporting may be controlled separately from PCI device Functions through the PCI Express Capability structure described in § Section 7.5.3 . The PCI-compatible error control and status register fields do not have any effect on PCI Express error reporting enabled through the PCI Express Capability structure. PCI Express device Functions may implement optional advanced error reporting as described in § Section 7.8.4 .

For PCI Express Root Ports represented by a Type 1 Configuration Space Header :

- The primary side Error Control/Status registers apply to errors detected on the internal logic associated with the Root Complex.
- The secondary side Error Control/Status registers apply to errors detected on the Link originating from the Root Port.

For PCI Express Switch Upstream Ports represented by a Type 1 Configuration Space Header :

- The primary side Error Control/Status registers apply to errors detected on the Upstream Link of the Switch.
- The secondary side Error Control/Status registers apply to errors detected on the internal logic of the Switch.

For PCI Express Switch Downstream Ports represented by a Type 1 Configuration Space Header :

- The primary side Error Control/Status registers apply to errors detected on the internal logic of the Switch.
- The secondary side Error Control/Status registers apply to errors detected on the Downstream Link originating from the Switch Port.

7.5.1.2 Type 0 Configuration Space Header §

§ Figure 7-10 details allocation for register fields of Type 0 Configuration Space Header for PCI Express device Functions.

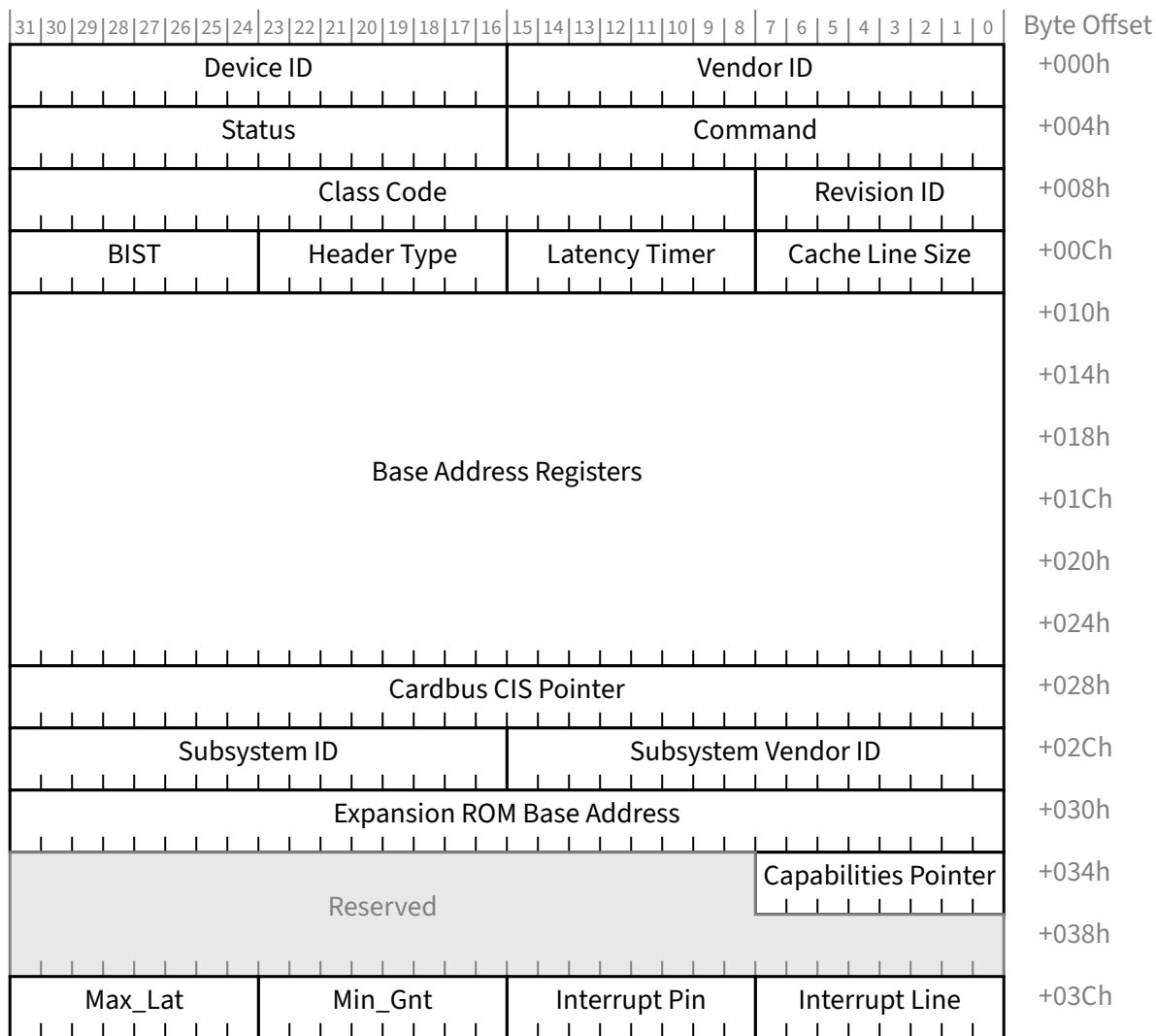


Figure 7-10 Type 0 Configuration Space Header §

§ Section 7.5.1.1 details the PCI Express-specific registers that are valid for all Configuration Space Header types. The PCI Express-specific interpretation of registers specific to Type 0 Configuration Space Header is defined in this section.

7.5.1.2.1 Base Address Registers (Offset 10h - 24h) §

System software must build a consistent address map before booting the machine to an operating system. This means it has to determine how much memory is in the system, and how much address space the Functions in the system require. After determining this information, system software can map the Functions into reasonable locations and proceed with

system boot. In order to do this mapping in a device-independent manner, the base registers for this mapping are placed in the predefined header portion of Configuration Space. It is strongly recommended that power-up firmware/software also support the optional Enhanced Configuration Access Mechanism (ECAM).

For VFs, these registers must be hardwired to Zero. See [§ Section 9.2.1.1.1](#) and [§ Section 9.4.3.14](#).

Bit 0 in all Base Address registers is read-only and used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b in bit 0 (see [§ Figure 7-11](#)). Base Address registers that map to I/O Space must return a 1b in bit 0 (see [§ Figure 7-12](#)).

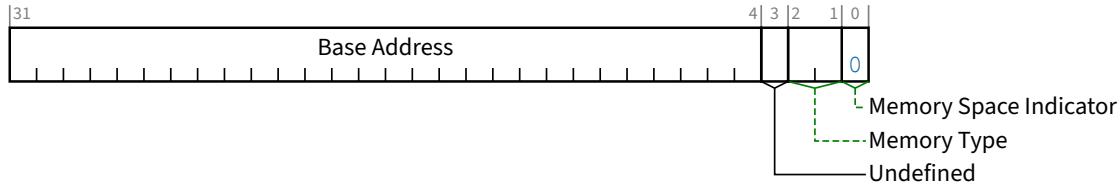


Figure 7-11 Base Address Register for Memory [§](#)

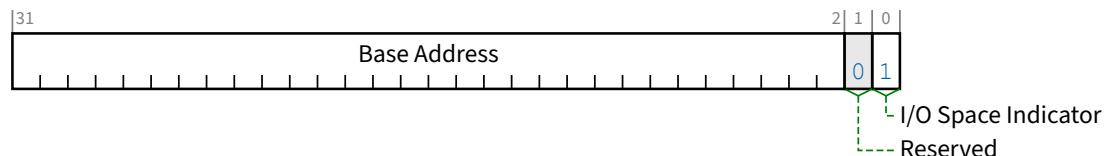


Figure 7-12 Base Address Register for I/O [§](#)

Base Address registers that map into I/O Space are always 32 bits wide with bit 0 hardwired to 1b. Bit 1 is Reserved and must return 0b on reads and the other bits are used to map the Function into I/O Space.

Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space) with bit 0 hardwired to 0b. It is strongly recommended for each Memory BAR in a PCI Express component to be 64 bits wide. Each Memory BAR in a PCI Express component is permitted to be 32 bits wide.

IMPLEMENTATION NOTE: GUIDANCE ON MEMORY BAR SIZES §

A key reason for making Memory BARs 64 bits wide is to avoid possible shortages of 32-bit MMIO space. Typical systems limit that space to 1 GB, and some support even less. If a 32-bit Memory BAR consumes 64 MB, only sixteen such BARs would consume the entire 1 GB of space, which is inadequate for some platforms. Memory BAR alignment constraints can further reduce the available space when different-sized Memory BARs are comingled in MMIO space.

A key reason for making Memory BARs 32 bits wide has been when a single Function requires greater than three MMIO ranges, since only three 64-bit BARs will fit inside the Base Address Registers area within a Type 0 Configuration Space Header. This led to some Functions using 32-bit Memory BARs. It is strongly recommended instead to combine multiple MMIO ranges under a single 64-bit BAR, properly aligned and separated to satisfy applicable processor page-size requirements.

Existing PCIe components may have used 32-bit Memory BARs for a variety of other reasons. It is strongly recommended that new designs use 64-bit Memory BARs wherever possible.

For Memory Base Address registers, bits 2 and 1 have an encoded meaning as shown in § Table 7-9 . The historical definition of bit 3 is now deprecated, and bit 3 is undefined. However, for backward compatibility with legacy system software, it is strongly recommended that bit 3 be Set for 64-bit Memory BARs and Clear for 32-bit Memory BARs. Bits 3-0 are read-only.

If a Host Bridge supports processor Memory Write byte merging¹⁷², this feature must be disabled unless mechanisms outside the scope of this specification prevent the feature from merging such writes that target any Function's Memory BAR range (regardless of the BAR's bit 3 value) that cannot tolerate them.

Table 7-9 Memory Base Address Register Bits 2:1 Encoding §

| Bits 2:1(b) | Meaning |
|-------------|--|
| 00 | Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space. |
| 01 | Reserved ¹⁷³ |
| 10 | Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space. |
| 11 | Reserved |

The number of upper bits that a Function actually implements depends on how much of the address space the Function will respond to. A 32-bit Base Address register supports a single, implementation specific, memory size that is a power of 2, from 16 bytes to 256 Bytes (I/O Space) or 128 Bytes to 2 GB (Memory Space). Each BAR must implement all appropriate upper address bits as Read-Write (i.e., a BAR must be able to be configured to any appropriately aligned address in the associated address space). A Function that wants a 1 MB memory address space (using a single 32-bit Base Address register) must implement the top 12 bits of the Address register as Read-Write, hardwiring the other address bits to 0's. The default value of Read-Write bits in BARs is implementation specific. The attributes for some of the bits in the BAR are affected by the Resizable BAR Capability, if it is implemented.

To determine how much address space a Function requires, system software should write a value of all 1's to each BAR register and then read the value back. Low-order bits of the Base Address field in each BAR must return 0's to indicate how much address space is required. Unimplemented Base Address registers must be hardwired to zero.

172. Refer to [PCI] for the definition and examples of byte merging.

173. The encoding to support memory space below 1 MB was supported in an earlier version of the PCI Local Bus Specification. System software should recognize this encoding and handle it appropriately.

This design implies that all address spaces used are a power of two in size and are naturally aligned. Functions are free to consume more address space than required, but decoding down to a 4 KB space for memory is suggested for Functions that need less than that amount. For instance, a Function that has 64 bytes of registers to be mapped into Memory Space may consume up to 4 KB of address space in order to minimize the number of bits in the address decoder. Functions that do consume more address space than they use are not required to respond to the unused portion of that address space if the Function's programming model never accesses the unused space. The Function is permitted to return Unsupported Request for accesses ~~↑targetting~~ ↑targetting the unused locations. Functions that map control functions into I/O Space must not consume more than 256 bytes per I/O Base Address register or per each entry in the Enhanced Allocation Capability . The upper 16 bits of the I/O Base Address register may be hardwired to zero for Functions intended for 16-bit I/O systems, such as PC compatibles. However, a full 32-bit decode of I/O addresses must still be done.

A Type 0 Configuration Space Header has six DWORD locations allocated for Base Address registers starting at offset 10h in Configuration Space. A Type 1 Configuration Space Header has only two DWORD locations. A Function may use any of the locations to implement Base Address registers. An implemented 64-bit Base Address register consumes two consecutive DWORD locations. Software looking for implemented Base Address registers must start at offset 10h and continue upwards through offset 24h. A typical Function requires one memory range for its control functions.

Some graphics Functions use two ranges, one for control functions and another for a frame buffer. A Function that wants to map control functions into both memory and I/O Spaces at the same time must implement two Base Address registers (one memory and one I/O) but such implementations are strongly discouraged in Functions that do not need to support legacy programming models. Depending on the operating system, the driver for that Function might only use one space in which case the other space will be unused. Functions are strongly recommended to always map control functions into Memory Space. When possible, it is strongly recommended for Functions not to request I/O Space resources.

IMPLEMENTATION NOTE: I/O BARS SHOULD BE AVOIDED WHEN POSSIBLE §

I/O Space is a limited resource in all systems. The use of such resources should be avoided when possible, and otherwise minimized.

Due to the 4-KB minimum granularity of any bridge's I/O forwarding range, a Function that requests the minimum 256 bytes in an I/O BAR will still force the system to allocate at least 4 KB of I/O Space to the bridge under which that Function resides. If a hierarchy domain contains multiple bridges, then the system may be forced to allocate even more I/O Space to satisfy a Function's I/O BAR(s).

Each Function has a limited number of BARs. Each implemented I/O BAR either reduces the number or size of possible Memory BARs. With a Legacy Endpoint, if the platform is unable to allocate I/O Space for any implemented I/O BARs, then the platform might not enable the Legacy Endpoint, even if the Legacy Endpoint can operate without allocated I/O Space. If any PCI Express Endpoints implement I/O BARs and the platform allocates I/O Space for them, this may contribute to a shortage of I/O Space for any Legacy Endpoints that are present. For these reasons, implementing I/O BARs should be avoided when possible.

The minimum Memory Space address range requested by a BAR is 128 bytes. The attributes for some of the bits in the BAR are affected by the Resizable BAR Capability, if it is implemented.

IMPLEMENTATION NOTE: ORGANIZATION OF MMIO SPACE BY ATTRIBUTES §

Processor architectures typically assign specific attributes to memory resources. These attributes impact the functionality and performance of memory operations and are outside the scope of this document. However, by following certain guidance in relation to these attributes, the interoperability and usability of PCI Express components can be significantly improved. This guidance applies to all situations, including but not limited to host processors, where code execution triggers the generation of Request TLPs and allows software/firmware developers the ability to control such aspects as sequencing, operation size, and address alignment. The concepts also apply to special-purpose hardware for generating Request TLPs, but it is generally assumed that the designers of such hardware have a more thorough control of Request TLP generation and understanding of Completer expectations, as compared to creators of software/firmware, where this is abstracted.

Processor memory attributes are typically assigned at the page level. This is the granularity at which software/firmware can reason about security properties and about a restricted programming model¹⁷⁴. Devices are strongly encouraged to organize MMIO space allocated via BARs as required to ensure that memory ranges with different access attribute requirements are located in different naturally aligned blocks, with a minimum block size of 4 KB and a recommended block size of at least 64 KB.

Some processor memory attributes may result in a TLP sequence that is closely correlated with the processor instruction sequence while other processor memory attributes may permit techniques like write combining, speculation of reads, and reordering of memory operations when creating TLPs from an instruction sequence.

Regardless of the attributes, it is strongly recommended that Read Side Effects are not implemented for any MMIO address unless the system aspects of such Read Side Effects are understood and managed.

Processor architectures are encouraged to clearly document the mechanisms for configuring memory attributes for MMIO ranges.

Device-specific software is expected to understand the device requirements of each range and choose an acceptable memory attribute. When device peer-to-peer operations are supported, the same considerations apply to the involved devices as to hosts.

Previous revisions of this specification indicated that access properties were correlated with the BAR register and address range chosen to define the MMIO space. Since TLPs must follow the same set of rules regardless of BAR programming it has always been permitted for host/device software/hardware to determine and use appropriate memory access attributes. Because 64-bit BARs provide significantly better ability to manage system MMIO resources, devices are strongly encouraged to use 64-bit BARs.

7.5.1.2.2 Cardbus CIS Pointer Register (Offset 28h) §

This register was originally described in the [PC-Card]. This register does not apply to PCI Express and must be hardwired to Zero.

7.5.1.2.3 Subsystem Vendor ID Register / Subsystem ID Register (Offset 2Ch/2Eh) §

The Subsystem Vendor ID and Subsystem ID registers are used to uniquely identify the adapter or subsystem where the PCI Express component resides. They provide a mechanism for vendors to distinguish their products from one another

¹⁷⁴ See “IMPLEMENTATION NOTE: OPTIMIZATIONS BASED ON RESTRICTED PROGRAMMING MODEL”

even though the assemblies may have the same PCI Express component on them (and, therefore, the same Vendor ID and Device ID).

Implementation of these registers is required for all Functions except those that have a Base Class 06h with Sub Class 00h-04h (00h, 01h, 02h, 03h, 04h), or a Base Class 08h with Sub Class 00h-03h (00h, 01h, 02h, 03h). Subsystem Vendor IDs can be obtained from the PCI SIG and are used to identify the vendor of the adapter, motherboard, or subsystem¹⁷⁵. A Subsystem Vendor ID (SVID) must be a Vendor ID assigned by the PCI-SIG to the vendor of the subsystem. In keeping with PCI-SIG procedures, valid vendor identifiers must be obtained from the PCI-SIG to ensure uniqueness.

Values for the Subsystem ID are vendor assigned. Subsystem ID values, in conjunction with the Subsystem Vendor ID, form a unique identifier for the PCI product. Subsystem ID and Device ID values are distinct and unrelated to each other, and software should not assume any relationship between them.

Values in these registers must be loaded before the Function becomes Configuration-Ready. How these values are loaded is not specified but could be done during the manufacturing process or loaded from external logic (e.g., strapping options, serial ROMs, etc.). These values must not be loaded using Expansion ROM software because Expansion ROM software is not guaranteed to be run during POST in all systems.

If a device is designed to be used exclusively on the system board, the system vendor may use system specific software to initialize these registers after each power-on.

The Subsystem Vendor ID register in a PF and its associated VFs must return the same value when read.

The Subsystem ID register in a PF and its associated VFs are permitted to return different values when read.

IMPLEMENTATION NOTE: SUBSYSTEM VENDOR ID AND SUBSYSTEM ID §

The Subsystem Vendor ID and Subsystem ID fields, taken together, allow software to uniquely identify a PCI circuit board product. Vendors should therefore not reuse Subsystem ID values across multiple product types that share a common Subsystem Vendor ID. It is acceptable, although not preferred, to reuse the Subsystem ID value on products with the same Subsystem Vendor ID in cases where the products are in the same family and generation, and differ only in capacity or performance. Note also that it is acceptable for vendors to use multiple unique Subsystem ID values over time for a single product type, such as to indicate some internal difference such as component selection.

7.5.1.2.4 Expansion ROM Base Address Register (Offset 30h) §

Some Functions, especially those that are intended for use on add-in cards, require local EPROMs for Expansion ROM (refer to the [Firmware] for a definition of ROM contents). This register is defined to handle the base address and size information for this Expansion ROM. The register layout is shown in § Figure 7-13 and § Table 7-10 describes the bits in the register.

For VFs, the Expansion ROM Base Address register is not supported and must be hardwired to Zero. The VI may choose to provide access to a PF's Expansion ROM Base Address register for its associated VFs via emulation.

Functions that support an expansion ROM must allow that ROM to be accessed with any combination of byte enables.

¹⁷⁵. A company requires only one Vendor ID. That value can be used in either the Vendor ID register of Configuration Space (e.g., offset 00h) or the Subsystem Vendor ID register of Configuration Space (e.g., offset 2Ch). It is used in the Vendor ID register (e.g., offset 00h) if the company built the silicon. It is used in the Subsystem Vendor ID register (e.g., offset 2Ch) if the company built the assembly. If a company builds both the silicon and the assembly, then the same value would be used in both registers.

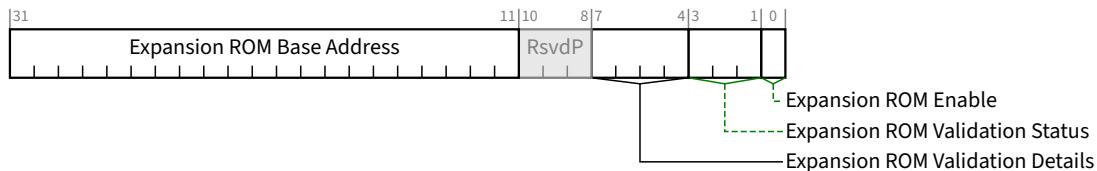


Figure 7-13 Expansion ROM Base Address Register §

Table 7-10 Expansion ROM Base Address Register §

| Bit Location | Description | Attributes |
|--------------|--|--------------|
| 0 | <p>Expansion ROM Enable - This bit controls whether or not the Function accepts accesses to its Expansion ROM via the Expansion ROM Base Address Register . Functions that support an Expansion ROM accessible through this register must implement this bit. If the Function has an Enhanced Allocation Capability that includes an EA entry for an Expansion ROM, this bit must be hardwired to 0b (see § Section 7.5.1.2.4). Functions that do not support an Expansion ROM are permitted to hardwire this bit to 0b. When this bit is 0b, the Function's Expansion ROM address space via the Expansion ROM Base Address Register is disabled. When the bit is 1b, address decoding is enabled using the Expansion ROM Base Address field in this register. This optionally allows a Function to be used with or without an Expansion ROM depending on system configuration. The Memory Space Enable bit in the Command register has precedence over the Expansion ROM Enable bit. A Function must claim accesses to its Expansion ROM via the Expansion ROM Base Address Register only if both the Memory Space Enable bit and the Expansion ROM Enable bit are Set. The default value of this bit is 0b.</p> <p>In order to minimize the number of address decoders needed, a Function is permitted to share a decoder between the Expansion ROM Base Address Register and other Base Address registers or entries in the Enhanced Allocation Capability¹⁷⁶. When Expansion ROM Enable is Set, the decoder is used for accesses to the Expansion ROM and device independent software must not access the Function through any other Base Address Registers or entries in the Enhanced Allocation Capability . Address decode sharing is not permitted for PFs or if the Function contains an Enhanced Allocation Capability with an EA entry for an Expansion ROM.</p> | RO / RW |
| 3:1 | <p>Expansion ROM Validation Status - Expansion ROM Validation is optional. When this field is non-zero, it indicates the status of hardware validation of the Expansion ROM contents.</p> <ul style="list-style-type: none"> An Expansion ROM is considered valid if it passes an implementation specific integrity check. An Expansion ROM is considered valid-warn if the implementation specific integrity check passes but indicates an implementation specific warning condition. A valid or valid-warn Expansion ROM is also considered trusted if passes an optional implementation specific trust test (e.g., signed by a trusted certificate). Hardware validation must include the contents of the Expansion ROM. This validation status is also permitted to cover additional internal information (e.g., internal firmware). Validation does not include Vital Product Data (see § Section 6.27). It is optional whether an implementation is capable of returning Validation Status values 011b, 101b, 110b, or 111b. <p>Defined encodings are:</p> <p>000b Validation not supported</p> | HwInit / ROS |

176. Note that it is the address decoder that is shared, not the registers themselves. The Expansion ROM Base Address Register and other Base Address Registers or entries in the Enhanced Allocation Capability must be able to hold unique values at the same time.

Base 6.4 vs Base 6.3

| Bit Location | Description | Attributes |
|--------------|---|---|
| | <p>001b Validation in Progress</p> <p>010b Validation Pass Valid contents, trust test was not performed</p> <p>011b Validation Pass Valid and trusted contents</p> <p>100b Validation Fail Invalid contents</p> <p>101b Validation Fail Valid but untrusted contents (e.g., Out of Date, Expired or Revoked Certificate)</p> <p>110b Warning Pass Validation Passed with implementation specific warning. Valid contents, trust test was not performed</p> <p>111b Warning Pass Validation Passed with implementation specific warning. Valid and trusted contents</p> <ul style="list-style-type: none"> If the Function does not support validation, this field must be hardwired to 000b. If the Function supports validation and has an Enhanced Allocation Capability with an EA entry for an Expansion ROM, this field is <u>HwInit</u> and its value must be between 010b and 111b (see § Section 7.8.5.3). Otherwise, this field is Read Only Sticky and has a default value of 001b. When validation completes, this field must contain a value between 010b and 111b inclusive. Software is permitted to assume validation will never complete if this field contains 001b and 1 minute has passed after de-assertion of Fundamental Reset. This field is only reset by Fundamental Reset, and is not affected by other resets. | |
| 7:4 | <p>Expansion ROM Validation Details - contains optional, implementation specific details associated with Expansion ROM Validation.</p> <ul style="list-style-type: none"> If the Function does not support validation, this field is <u>RsvdP</u>. This field is optional. When validation is supported and this field is not implemented, this field must be hardwired to 0000b. Any unused bits in this field are permitted to be hardwired to 0b. If validation is in progress (Expansion ROM Validation Status is 001b), non-zero values of this field represent implementation specific indications of the phase of the validation progress (e.g., 50% complete). The value 0000b indicates that no validation progress information is provided. If validation is completed (Expansion ROM Validation Status 010b to 111b inclusive), non-zero values in this field represent additional implementation specific information. The value 0000b indicates that no information is provided. If the Function supports validation and has an Enhanced Allocation Capability with an EA entry for an Expansion ROM, this field is <u>HwInit</u>. Otherwise, this field is Read Only Sticky. This field is only reset by Fundamental Reset, and is not affected by other resets. This field must not change value once the validation process completes. It is recommended that system software include the value of this field when it reports validation status (e.g., error log). | <u>HwInit</u> / <u>ROS</u> / <u>RsvdP</u> |
| 31:11 | <p>Expansion ROM Base Address - contains the upper bits 21 bits of the starting memory address of the Expansion ROM. The lower 11 bits of the <u>Expansion ROM Base Address Register</u> are masked off (set to zero) by software to form a 32-bit address.</p> <p>This field functions like the address portion of a 32-bit Base Address register. This field corresponds to the upper 21 bits of the <u>Expansion ROM Base Address</u>. The number of bits (out of these 21) that a Function actually implements depends on how much Expansion ROM address space the Function requires. For instance, a Function that requires a 64 KB area to map its Expansion ROM would implement the top 16</p> | <u>RW</u> / <u>RO</u> |

| Bit Location | Description | Attributes |
|--------------|--|------------|
| | <p>bits in this field as writeable, leaving the bottom 5 bits (out of these 21) hardwired to 0b. The amount of address space a Function requests must not be greater than 16 MB. Functions that support an Expansion ROM accessible through this register must implement this field. If the Function has an Enhanced Allocation Capability that includes an EA entry for an Expansion ROM, this field must be hardwired to 0 (see § Section 7.8.5.3) Functions that do not support an Expansion ROM are permitted to hardwire this field to 0.</p> <p>Device independent configuration software can determine how much address space the Function requires by writing a value of all 1's to this field and then reading the value back. Low order bits of this field must return 0's in all don't-care bits, effectively specifying the size and alignment requirements. The amount of address space a Function requests must not be greater than 16 MB.</p> | |

7.5.1.2.5 *Min_Gnt Register / Max_Lat Register (Offset 3Eh/3Fh)* §

These registers do not apply to PCI Express and must be hardwired to Zero.

7.5.1.3 *Type 1 Configuration Space Header* §

§ Figure 7-14 details allocation for register fields of Type 1 Configuration Space Header for Switch and Root Ports.

Base 6.4 vs Base 6.3

| | |
|---|----------------------|
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | Byte Offset |
| Device ID | +000h |
| Status | +004h |
| Class Code | +008h |
| BIST | +00Ch |
| Header Type | +010h |
| Primary Latency Timer | +014h |
| Cache Line Size | +018h |
| Base Address Register 0 | +01Ch |
| Base Address Register 1 | +020h |
| Secondary Latency Timer | +024h |
| Subordinate Bus Number | +028h |
| Secondary Bus Number | +02Ch |
| Primary Bus Number | +030h |
| Secondary Status | +034h |
| I/O Limit | +038h |
| I/O Base | +03Ch |
| Memory Limit | |
| 64-bit Memory Limit | |
| 64-bit Base Upper 32 Bits | |
| 64-bit Limit Upper 32 Bits | |
| I/O Base Limit 16 Bits | |
| I/O Base Upper 16 Bits | |
| Reserved | Capabilities Pointer |
| Expansion ROM Base Address | |
| Bridge Control | |
| Interrupt Pin | |
| Interrupt Line | |

Figure 7-14 Type 1 Configuration Space Header [§](#)

§ Section 7.5.1.1 details the PCI Express-specific registers that are valid for all Configuration Space Header types. The PCI Express-specific interpretation of registers specific to Type 1 Configuration Space Header is defined in this section.

Register interpretations described in this section apply to PCI-PCI Bridge structures representing Switch and Root Ports; other device Functions such as PCI Express to PCI/PCI-X Bridges with Type 1 Configuration Space headers are not covered by this section.

TLP routing is determined by the following registers, along with FPB (if implemented):

- Primary Bus Number
- Secondary Bus Number
- Subordinate Bus Number
- I/O Base
- I/O Limit

- Memory Base
- Memory Limit
- 64-bit Memory Base
- 64-bit Memory Limit
- 64-bit Base Upper 32 Bits
- 64-bit Limit Upper 32 Bits
- I/O Base Upper 16 Bits
- I/O Base Limit 16 Bits

7.5.1.3.1 Type 1 Base Address Registers (Offset 10h-14h) §

These registers are defined in § Section 7.5.1.2.1 . However the number of BARs available within the Type 1 Configuration Space Header is different than that of the Type 0 Configuration Space Header .

7.5.1.3.2 Primary Bus Number Register (Offset 18h) §

Except as noted, this register is not used by PCI Express Functions but must be implemented as read-write and the default value must be 00h, for compatibility with legacy software. PCI Express Functions capture the Bus (and Device) Number as described (including exceptions) in § Section 2.2.6 . Refer to [PCIe-to-PCI-PCI-X-Bridge] for exceptions to this requirement.

7.5.1.3.3 Secondary Bus Number Register (Offset 19h) §

The Secondary Bus Number register is used to record the bus number of the PCI bus segment to which the secondary interface of the bridge is connected. Configuration software programs the value in this register. The Bridge uses this register to determine when and how to respond to an ID-routed TLP observed on its primary interface, notably when to forward the TLP to its secondary interface, in certain cases after performing some conversion. See § Section 7.3.3 for Configuration Request routing and conversion rules. This register must be implemented as read/write and the default value must be 00h.

7.5.1.3.4 Subordinate Bus Number Register (Offset 1Ah) §

The Subordinate Bus Number register is used to record the bus number of the highest numbered PCI bus segment which is behind (or subordinate to) the bridge. Configuration software programs the value in this register. The Bridge uses this register to determine when and how to respond to an ID-routed TLP observed on its primary interface, notably when to forward the TLP to its secondary interface. See § Section 7.3.3 for Configuration Request routing rules. This register must be implemented as read-write and the default value must be 00h.

7.5.1.3.5 Secondary Latency Timer (Offset 1Bh) §

This register does not apply to PCI Express. It must be read-only and hardwired to 00h. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register.

7.5.1.3.6 I/O Base / I/O Limit Registers(Offset 1Ch/1Dh) §

The I/O Base and I/O Limit registers are optional and define an address range that is used by the bridge to determine when to forward I/O transactions from one interface to the other.

If a bridge does not implement an I/O address range, then both the I/O Base and I/O Limit registers must be implemented as read-only registers that return zero when read. If a bridge supports an I/O address range, then these registers must be initialized by configuration software so default states are not specified.

If a bridge implements an I/O address range, the upper 4 bits of both the I/O Base and I/O Limit registers are writable and correspond to address bits Address[15:12]. For the purpose of address decoding, the bridge assumes that the lower 12 address bits, Address[11:0], of the I/O base address (not implemented in the I/O Base register) are zero. Similarly, the bridge assumes that the lower 12 address bits, Address[11:0], of the I/O limit address (not implemented in the I/O Limit register) are FFFh. Thus, the bottom of the defined I/O address range will be aligned to a 4 KB boundary and the top of the defined I/O address range will be one less than a 4 KB boundary.

The I/O Limit register can be programmed to a smaller value than the I/O Base register, if there are no I/O addresses on the secondary side of the bridge. In this case, the bridge will not forward any I/O transactions from the primary bus to the secondary and will forward all I/O transactions from the secondary bus to the primary bus.

The lower four bits of both the I/O Base and I/O Limit registers are read-only, contain the same value, and encode the I/O addressing capability of the bridge according to § Table 7-11 .

Table 7-11 I/O Addressing Capability §

| Bits 3:0 | I/O Addressing Capability |
|----------|---------------------------|
| 0h | 16-bit I/O addressing |
| 1h | 32-bit I/O addressing |
| 2h-Fh | Reserved |

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

If the low four bits of the I/O Base and I/O Limit registers are 0001b, then the bridge supports 32-bit I/O address decoding, and the I/O Base Upper 16 Bits and the I/O Limit Upper 16 Bits hold the upper 16 bits, corresponding to Address[31:16], of the 32-bit I/O Base and I/O Limit addresses respectively. In this case, system configuration software is permitted to locate the I/O address range supported by the bridge anywhere in the 4 GB I/O Space. Note that the 4 KB alignment and granularity restrictions still apply when the bridge supports 32-bit I/O addressing.

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

7.5.1.3.7 Secondary Status Register (Offset 1Eh) §

§ Table 7-12 defines the Secondary Status Register and § Figure 7-15 provides the register layout. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register.

Base 6.4 vs Base 6.3

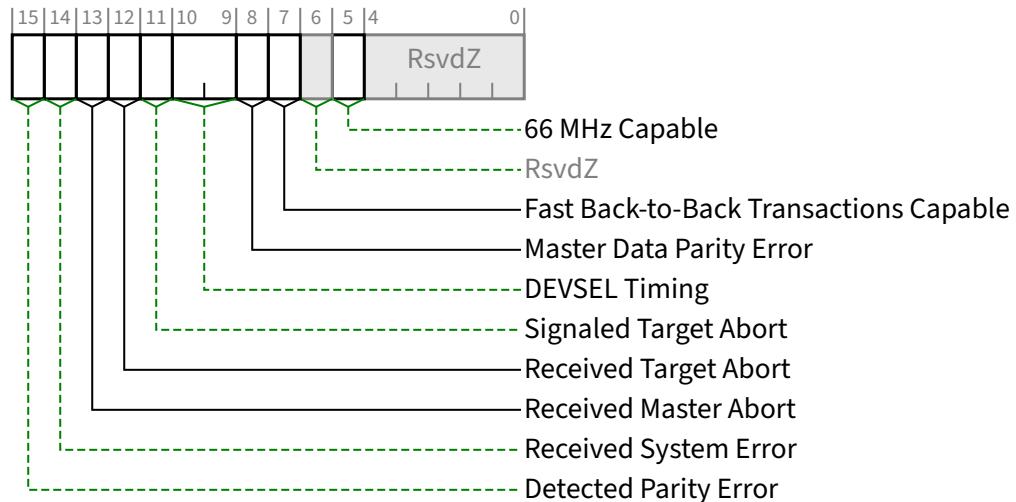


Figure 7-15 Secondary Status Register §

Table 7-12 Secondary Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 5 | 66 MHz Capable - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 7 | Fast Back-to-Back Transactions Capable - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 8 | <p>Master Data Parity Error - See § Section 7.5.1.1.14.</p> <p>This bit is Set by a Function with a Type 1 Configuration Space Header if the Parity Error Response Enable bit in the Bridge Control Register is Set and either of the following two conditions occurs:</p> <ul style="list-style-type: none"> Port receives a Poisoned Completion coming Upstream Port transmits a Poisoned Request Downstream <p>If the Parity Error Response Enable bit is Clear, this bit is never Set.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 10:9 | DEVSEL Timing - This field was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the field must be hardwired to 00b. | RO |
| 11 | <p>Signaled Target Abort - See § Section 7.5.1.1.14.</p> <p>This bit is Set when the Secondary Side for Type 1 Configuration Space Header Function (for Requests completed by the Type 1 header Function itself) completes a Posted or Non-Posted Request as a Completer Abort error.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 12 | <p>Received Target Abort - See § Section 7.5.1.1.14.</p> <p>This bit is Set when the Secondary Side for Type 1 Configuration Space Header Function (for Requests initiated by the Type 1 header Function itself) receives a Completion with Completer Abort Completion Status.</p> | RW1C |

| Bit Location | Register Description | Attributes |
|--------------|---|-------------|
| | Default value of this bit is 0b. | |
| 13 | <p>Received Master Abort - See § Section 7.5.1.1.14.</p> <p>This bit is Set when the Secondary Side for Type 1 Configuration Space Header Function (for Requests initiated by the Type 1 header Function itself) receives a Completion with Unsupported Request Completion Status.</p> <p>Default value of this bit is 0b.</p> | <u>RW1C</u> |
| 14 | <p>Received System Error - See § Section 7.5.1.1.14.</p> <p>This bit is Set when the Secondary Side for a Type 1 Configuration Space Header Function receives an ERR_FATAL or ERR_NONFATAL Message.</p> <p>Default value of this bit is 0b.</p> | <u>RW1C</u> |
| 15 | <p>Detected Parity Error - See § Section 7.5.1.1.14.</p> <p>This bit is Set by a Function with a Type 1 Configuration Space Header when a Poisoned TLP is received by its Secondary Side, regardless of the state the Parity Error Response Enable bit in the Bridge Control Register.</p> <p>Default value of this bit is 0b.</p> | <u>RW1C</u> |

7.5.1.3.8 Memory Base Register / Memory Limit Register (Offset 20h/22h) [§](#)

The Memory Base and Memory Limit registers define a memory mapped address range which is used by the bridge to determine when to forward memory transactions from one interface to the other (see the [PCI-to-PCI-Bridge] for additional details).

The upper 12 bits of both the Memory Base and Memory Limit registers are read/write and correspond to the upper 12 address bits, Address[31:20], of 32-bit addresses. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, Address[19:0], of the memory base address (not implemented in the Memory Base register) are zero. Similarly, the bridge assumes that the lower 20 address bits, Address[19:0], of the memory limit address (not implemented in the Memory Limit register) are F FFFFh. Thus, the bottom of the defined memory address range will be aligned to a 1 MB boundary and the top of the defined memory address range will be one less than a 1 MB boundary.

The Memory Limit register must be programmed to a smaller value than the Memory Base register if there is no memory-mapped address space on the secondary side of the bridge.

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

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

7.5.1.3.9 64-bit Memory Base / 64-bit Memory Limit Registers (Offset 24h/26h) and 64-bit Base Upper 32 Bits / 64-bit Limit Upper 32 Bits Registers (Offset 28h/2Ch) [§](#)

The 64-bit Memory Base and 64-bit Memory Limit registers define a memory address range that is used by the bridge to determine when to forward memory transactions from one interface to the other.

If not implemented, both the 64-bit Memory Base and 64-bit Memory Limit registers must be read-only and return zero when read.

If the bridge implements these registers, the upper 12 bits of each register are read/write and correspond to Address[31:20], of a 64-bit address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, Address[19:0], of the 64-bit base address are zero. Similarly, the bridge assumes that the lower 20 address bits, Address[19:0], of the 64-bit limit address are F FFFFh. Thus, the bottom of the memory address range will be aligned to a 1 MB boundary and the top of the defined memory address range will be one less than a 1 MB boundary.

The 64-bit Base Upper 32 Bits and 64-bit Limit Upper 32 Bits registers (defined below) correspond to Address[63:32] of the Base and Limit addresses, respectively.

If it is intended that these registers not be used to indicate memory mapped to the secondary side of the bridge, System Software must program the 64-bit Memory Limit register to a smaller value than the 64-bit Memory Base register.

The bottom 4 bits of both the 64-bit Memory Base and 64-bit Memory Limit registers must be read-only, contain the same value, and encode whether or not the bridge supports 64-bit addresses. If these four bits have the value 0h, then the 64-bit Base/Limit Upper 32 registers are ignored and treated as containing all zeros. If these four bits have the value 01h, then the bridge supports 64-bit addresses and the 64-bit Base Upper 32 Bits and 64-bit Limit Upper 32 Bits registers hold the rest of the 64-bit base and limit addresses respectively. All other encodings are Reserved.

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

7.5.1.3.10 64-bit Base Upper 32 Bits / 64-bit Limit Upper 32 Bits Registers (Offset 28h/2Ch) §

If not implemented, then both 64-bit Base Upper 32 Bits and 64-bit Limit Upper 32 Bits registers must be read-only and return zero when read. If a bridge implements these registers, then both of these registers must be implemented as read/write registers that must be initialized by configuration software. They specify the upper 32 bits, corresponding to Address[63:32], of the 64-bit base and limit addresses.

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

7.5.1.3.11 I/O Base Upper 16 Bits / I/O Limit Upper 16 Bits Registers (Offset 30h/32h) §

The I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are optional extensions to the I/O Base and I/O Limit registers. If the I/O Base and I/O Limit registers indicate support for 16-bit I/O address decoding, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are implemented as read-only registers which return zero when read.

If the I/O Base and I/O Limit registers indicate support for 32-bit I/O addressing, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers must be initialized by configuration software so default states are not specified.

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

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

7.5.1.3.12 Expansion ROM Base Address Register (Offset 38h) §

This register is defined in § Section 7.5.1.2.4 . However the offset of the register within the Type 1 Configuration Space Header is different than that of the Type 0 Configuration Space Header.

7.5.1.3.13 Bridge Control Register (Offset 3Eh) §

The Bridge Control Register provides extensions to the Command Register that are specific to a Function with a Type 1 Configuration Space Header . The Bridge Control Register provides many of the same controls for the secondary interface that are provided by the Command Register for the primary interface. There are some bits that affect the operation of both interfaces of the bridge.

§ Table 7-13 defines the Bridge Control Register and § Figure 7-16 depicts register layout. For PCI Express to PCI/PCI-X Bridges, refer to the [PCIe-to-PCI-PCI-X-Bridge] for requirements for this register.

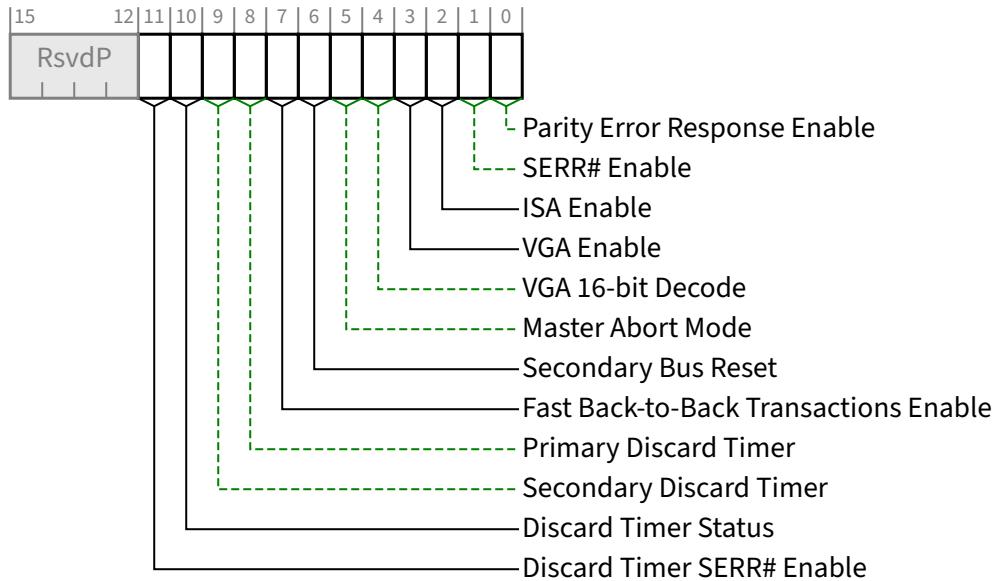


Figure 7-16 Bridge Control Register §

Table 7-13 Bridge Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Parity Error Response Enable - See § Section 7.5.1.1.14 .</p> <p>This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Secondary Status Register .</p> <p>Default value of this bit is 0b.</p> | RW |
| 1 | <p>SERR# Enable - See § Section 7.5.1.1.14 .</p> <p>This bit controls forwarding of <u>ERR_COR</u>, <u>ERR_NONFATAL</u> and <u>ERR_FATAL</u> from secondary to primary.</p> <p>Default value of this bit is 0b.</p> | RW |
| 2 | <p>ISA Enable - Modifies the response by the bridge to ISA I/O addresses. This applies only to I/O addresses that are enabled by the I/O Base and I/O Limit registers and are in the first 64 KB of I/O address space (0000 0000h to 0000 FFFFh). If this bit is Set, the bridge will block any forwarding from primary to secondary of I/O transactions addressing the last 768 bytes in each 1 KB block. In the opposite direction</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>(secondary to primary), I/O transactions will be forwarded if they address the last 768 bytes in each 1 KB block.</p> <p>0b forward downstream all I/O addresses in the address range defined by the <u>I/O Base</u> and <u>I/O Limit registers</u></p> <p>1b forward upstream ISA I/O addresses in the address range defined by the <u>I/O Base</u> and <u>I/O Limit registers</u> that are in the first 64 KB of PCI I/O address space (top 768 bytes of each 1 KB block)</p> <p>Default value of this bit is 0b.</p> | |
| 3 | <p>VGA Enable - Modifies the response by the bridge to VGA compatible addresses. If the <u>VGA Enable</u> bit is Set, the bridge will positively decode and forward the following accesses on the primary interface to the secondary interface (and, conversely, block the forwarding of these addresses from the secondary to primary interface):</p> <ul style="list-style-type: none"> Memory accesses in the range 000A 0000h to 000B FFFFh I/O addresses in the first 64 KB of the I/O address space (Address[31:16] are 0000h) where Address[9:0] are in the ranges 3B0h to 3BBh and 3C0h to 3DFh (inclusive of ISA address aliases determined by the setting of VGA 16-bit Decode) <p>If the <u>VGA Enable</u> bit is Set, forwarding of these accesses is independent of the <u>I/O address range</u> and <u>memory address ranges</u> defined by the <u>I/O Base</u> and <u>Limit registers</u>, the <u>Memory Base</u> and <u>64-bit Memory Limit registers</u> of the bridge. Forwarding of these accesses is also independent of the setting of the <u>ISA Enable</u> bit (in the <u>Bridge Control Register</u>) when the <u>VGA Enable</u> bit is Set. Forwarding of these accesses is qualified by the <u>I/O Space Enable</u> and <u>Memory Space Enable</u> bits in the <u>Command Register</u>.</p> <p>0b do not forward VGA compatible memory and I/O addresses from the primary to the secondary interface (addresses defined above) unless they are enabled for forwarding by the defined I/O and memory address ranges</p> <p>1b forward VGA compatible memory and I/O addresses (addresses defined above) from the primary interface to the secondary interface (if the <u>I/O Space Enable</u> and <u>Memory Space Enable</u> bits are set) independent of the I/O and memory address ranges and independent of the <u>ISA Enable</u> bit</p> <p>Functions that do not support VGA are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW |
| 4 | <p>VGA 16-bit Decode - This bit only has meaning if bit 3 (<u>VGA Enable</u>) of this register is also Set, enabling VGA I/O decoding and forwarding by the bridge.</p> <p>This bit enables system configuration software to select between 10-bit and 16-bit I/O address decoding for all VGA I/O register accesses that are forwarded from primary to secondary.</p> <p>0b execute 10-bit address decodes on VGA I/O accesses</p> <p>1b execute 16-bit address decodes on VGA I/O accesses</p> <p>Functions that do not support VGA are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW |
| 5 | <p>Master Abort Mode - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.</p> | RO |
| 6 | <p>Secondary Bus Reset - Setting this bit triggers a Hot Reset on the Secondary Side PCI Express Port. Software must ensure a minimum reset duration of 1 ms, (corresponding to T_{rst} in [PCI]). Software and systems must honor first-access-following-reset timing requirements defined in § <u>Section 6.6</u>, unless the Readiness Notifications mechanism (see § <u>Section 6.22</u>) is used or if the <u>Immediate Readiness</u> bit in the relevant Function's Status register is Set.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | Port configuration registers must not be changed, except as required to update Port status. See Implementation Note: Delays in Data Link Layer Link Active Reflecting Link Control Operations for related information. Default value of this bit is 0b. | |
| 7 | Fast Back-to-Back Transactions Enable - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 8 | Primary Discard Timer - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 9 | Secondary Discard Timer - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 10 | Discard Timer Status - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. | RO |
| 11 | Discard Timer SERR# Enable - This bit was originally described in the [PCI-to-PCI-Bridge]. Its functionality does not apply to PCI Express and must be hardwired to 0b. | RO |

7.5.2 PCI Power Management Capability Structure §

This section describes the registers making up the PCI Power Management Interface structure.

§ Figure 7-17 illustrates the organization of the PCI Power Management Capability structure. This structure is required for all non-VF PCI Express Functions. It is optional for VFs.

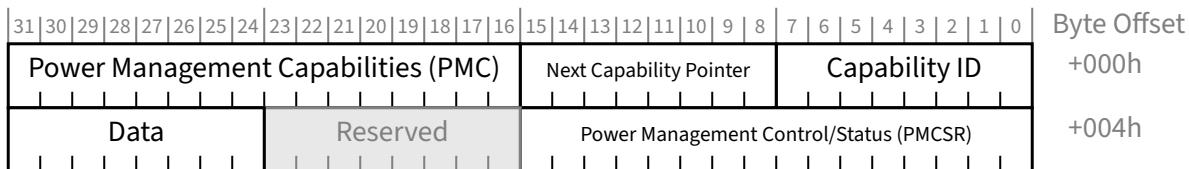


Figure 7-17 PCI Power Management Capability Structure §

Note: The 8-bit Power Management Data Register (Offset 07h) is optional for both Type 0 and Type 1 Functions.

PCI Express device Functions are required to support D0 and D3 device states; PCI-PCI Bridge structures representing PCI Express Ports as described in § Section 7.1 are required to indicate PME Message passing capability due to the in-band nature of PME messaging for PCI Express.

The PME_Status bit for the PCI-PCI Bridge structure representing PCI Express Ports, however, is only Set when the PCI-PCI Bridge Function is itself generating a PME. The PME_Status bit is not Set when the Bridge is propagating a PME Message but the PCI-PCI Bridge Function itself is not internally generating a PME.

7.5.2.1 Power Management Capabilities Register (Offset 00h) §

§ Figure 7-18 details allocation of register fields for Power Management Capabilities Register and § Table 7-14 describes the requirements for this register.

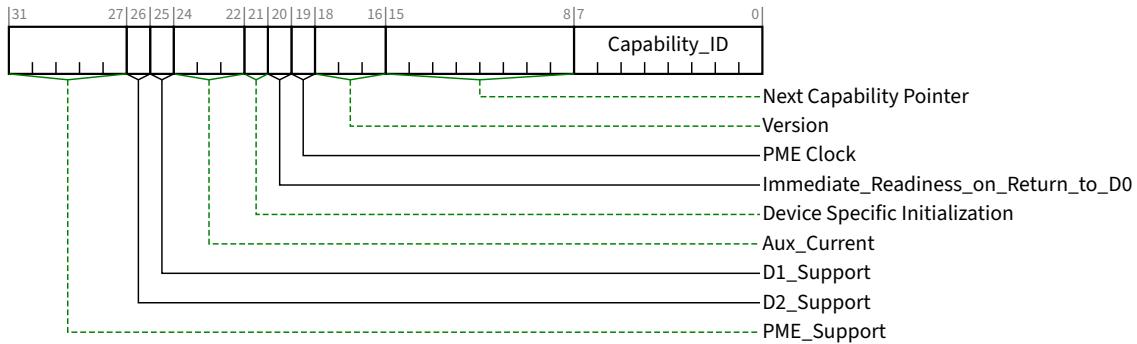


Figure 7-18 Power Management Capabilities Register §

Table 7-14 Power Management Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Capability_ID - This field returns 01h to indicate that this is the PCI Power Management Capability . Each Function may have only one item in its capability list with <u>Capability_ID</u> set to 01h. | RO |
| 15:8 | Next Capability Pointer - This field provides an offset into the Function's Configuration Space pointing to the location of next item in the capabilities list. If there are no additional items in the capabilities list, this field is set to 00h. | RO |
| 18:16 | Version - Must be hardwired to 011b for Functions compliant to this specification. | RO |
| 19 | PME Clock - Does not apply to PCI Express and must be hardwired to 0b. | RO |
| 20 | Immediate_Readiness_on_Return_to_D0 - If this bit is a “1”, this Function is guaranteed to be ready to successfully complete valid accesses immediately after being set to D0 . These accesses include Configuration cycles, and if the Function returns to <u>D0active</u> , they also include Memory and I/O Cycles. When this bit is “1”, for accesses to this Function, software is exempt from all requirements to delay accesses following a transition to D0 , including but not limited to the 10 ms delay; the delays described in § Section 5.9 . How this guarantee is established is beyond the scope of this document. It is permitted that system software/firmware provide mechanisms that supersede the indication provided by this bit, however such software/firmware mechanisms are outside the scope of this specification. | RO |
| 21 | Device Specific Initialization - The DSI bit indicates whether special initialization of this Function is required. When Set indicates that the Function requires a device specific initialization sequence following a transition to the <u>D0uninitialized</u> state. | RO |
| 24:22 | Aux_Current ¹⁷⁷ - This 3 bit field reports the <u>Vaux</u> auxiliary current requirements for the Function. | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|------------------------|---|------------------------|-------------------------------------|------------------------|-------------------------------------|------------------------|-------------------------------------|------------------------|---|------------------------|--|-------------|--------------------------------|-------------|-------------------------------|-------------|--------------------------------------|--|
| | <p>If this Function implements the Power Management Data Register, this field must be hardwired to 000b.</p> <p>If <u>PME_Support</u> is 0 xxxx b (PME assertion from <u>D3_Cold</u> is not supported) and the <u>Aux Power PM Enable</u> feature is not implemented, this field must be hardwired to 000b.</p> <p>For Functions where <u>PME_Support</u> is 1 xxxx b (PME assertion from <u>D3_Cold</u> is supported), and which do not implement the Power Management Data Register, the following encodings apply :</p> <p>Encoding Vaux Max. Power Required</p> <table> <tr> <td>111b</td> <td>1238 mW (e.g., 3.3 V at 375 mA)</td> </tr> <tr> <td>110b</td> <td>1056 mW (e.g., 3.3 V at 320 mA)</td> </tr> <tr> <td>101b</td> <td>891 mW (e.g., 3.3 V at 270 mA)</td> </tr> <tr> <td>100b</td> <td>726 mW (e.g., 3.3 V at 220 mA)</td> </tr> <tr> <td>011b</td> <td>528 mW (e.g., 3.3 V at 160 mA)</td> </tr> <tr> <td>010b</td> <td>330 mW (e.g., 3.3 V at 100 mA)</td> </tr> <tr> <td>001b</td> <td>182 mW (e.g., 3.3 V at 55 mA)</td> </tr> <tr> <td>000b</td> <td>0 mW (no Vaux power or self powered)</td> </tr> </table> <p>For encoding 000b, when the add-in card is self powered (e.g., it contains a battery), it is recommended that the <u>Power Budgeting Extended Capability</u> be used to report the thermal requirements of the add-in card.</p> <p>Note: Additional Aux power is permitted to be allocated using the firmware based mechanism (see the Request D3_Cold Aux Power Limit _DSM call as defined in [Firmware]). Additional Aux power is also permitted to be allocated by selecting a PM Sub State in the Power Limit mechanism (see § <u>Section 7.8.1.3</u>).</p> | 111b | 1238 mW (e.g., 3.3 V at 375 mA) | 110b | 1056 mW (e.g., 3.3 V at 320 mA) | 101b | 891 mW (e.g., 3.3 V at 270 mA) | 100b | 726 mW (e.g., 3.3 V at 220 mA) | 011b | 528 mW (e.g., 3.3 V at 160 mA) | 010b | 330 mW (e.g., 3.3 V at 100 mA) | 001b | 182 mW (e.g., 3.3 V at 55 mA) | 000b | 0 mW (no Vaux power or self powered) | |
| 111b | 1238 mW (e.g., 3.3 V at 375 mA) | | | | | | | | | | | | | | | | | |
| 110b | 1056 mW (e.g., 3.3 V at 320 mA) | | | | | | | | | | | | | | | | | |
| 101b | 891 mW (e.g., 3.3 V at 270 mA) | | | | | | | | | | | | | | | | | |
| 100b | 726 mW (e.g., 3.3 V at 220 mA) | | | | | | | | | | | | | | | | | |
| 011b | 528 mW (e.g., 3.3 V at 160 mA) | | | | | | | | | | | | | | | | | |
| 010b | 330 mW (e.g., 3.3 V at 100 mA) | | | | | | | | | | | | | | | | | |
| 001b | 182 mW (e.g., 3.3 V at 55 mA) | | | | | | | | | | | | | | | | | |
| 000b | 0 mW (no Vaux power or self powered) | | | | | | | | | | | | | | | | | |
| 25 | D1_Support - If this bit is Set, this Function supports the <u>D1</u> Power Management State. Functions that do not support <u>D1</u> must always return a value of 0b for this bit. | RO | | | | | | | | | | | | | | | | |
| 26 | D2_Support - If this bit is Set, this Function supports the <u>D2</u> Power Management State. Functions that do not support <u>D2</u> must always return a value of 0b for this bit. | RO | | | | | | | | | | | | | | | | |
| 31:27 | <p>PME_Support - This 5-bit field indicates the power states in which the Function may generate a PME and/or forward PME Messages.</p> <p>A value of 0b for any bit indicates that the Function is not capable of asserting PME while in that power state.</p> <table> <tr> <td>bit(27) X XXX1b</td> <td>PME can be generated from <u>D0</u></td> </tr> <tr> <td>bit(28) X XX1Xb</td> <td>PME can be generated from <u>D1</u></td> </tr> <tr> <td>bit(29) X X1XXb</td> <td>PME can be generated from <u>D2</u></td> </tr> <tr> <td>bit(30) X 1XXXb</td> <td>PME can be generated from <u>D3_Hot</u></td> </tr> <tr> <td>bit(31) 1 XXXXb</td> <td>PME can be generated from <u>D3_Cold</u></td> </tr> </table> <p>Bit 31 (PME can be asserted from <u>D3_Cold</u>) represents a special case. Functions that Set this bit require some sort of auxiliary power source. Implementation specific mechanisms are recommended to validate that the power source is available before setting this bit.</p> <p>Each bit that corresponds to a supported D-state must be Set for PCI-PCI Bridge structures representing Ports on Root Complexes/Switches to indicate that the Bridge will forward PME Messages. Bit 31 must only be Set if the Port is still able to forward PME Messages when main power is not available.</p> | bit(27) X XXX1b | PME can be generated from <u>D0</u> | bit(28) X XX1Xb | PME can be generated from <u>D1</u> | bit(29) X X1XXb | PME can be generated from <u>D2</u> | bit(30) X 1XXXb | PME can be generated from <u>D3_Hot</u> | bit(31) 1 XXXXb | PME can be generated from <u>D3_Cold</u> | RO | | | | | | |
| bit(27) X XXX1b | PME can be generated from <u>D0</u> | | | | | | | | | | | | | | | | | |
| bit(28) X XX1Xb | PME can be generated from <u>D1</u> | | | | | | | | | | | | | | | | | |
| bit(29) X X1XXb | PME can be generated from <u>D2</u> | | | | | | | | | | | | | | | | | |
| bit(30) X 1XXXb | PME can be generated from <u>D3_Hot</u> | | | | | | | | | | | | | | | | | |
| bit(31) 1 XXXXb | PME can be generated from <u>D3_Cold</u> | | | | | | | | | | | | | | | | | |

177. Earlier versions of this specification defined power levels as current (mA) at 3.3 V (hence the field name “Aux_Current”). To support form factors with different Aux Power voltage levels, the definition was changed to the equivalent wattage values (mW).

7.5.2.2 Power Management Control/Status Register (Offset 04h) §

This register is used to manage the PCI Function's power management state as well as to enable/monitor PMEs.

The PME Context includes the value of the PME_Status and PME_En bits, implementation specific state needed during D3Cold to implement the wakeup functionality (e.g., recognized a Wake on LAN packet and generate a PME Message), as well as any additional implementation specific state that must be preserved during a transition to the D0uninitialized state.

If a Function supports PME generation from D3Cold, its PME Context is not affected by Reset. This is because the Function's PME functionality itself may have been responsible for the wake event which caused the transition back to D0. Therefore, the PME Context must be preserved for the system software to process.

If PME generation is not supported from D3Cold, then all PME Context is initialized with the assertion of Reset.

§ Figure 7-19 details allocation of the register fields for the Power Management Control/Status Register and § Table 7-15 describes the requirements for this register.

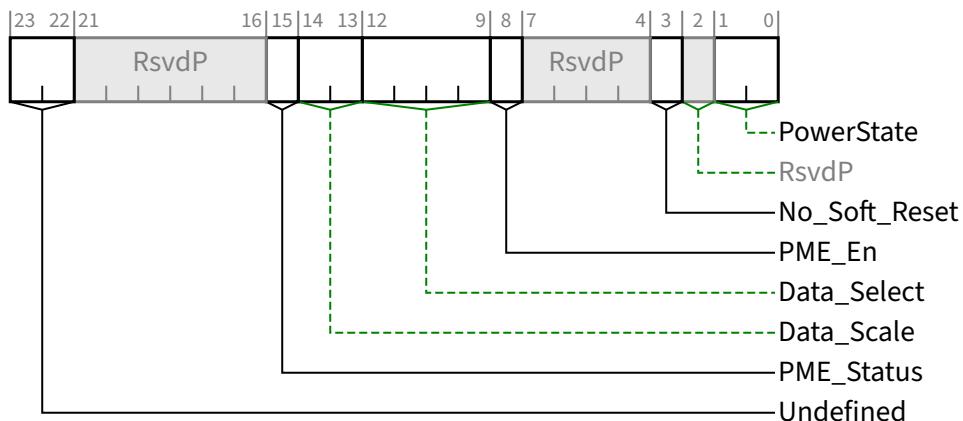


Figure 7-19 Power Management Control/Status Register §

Table 7-15 Power Management Control/Status Register §

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|---|------------|----|------------|----|------------|----|------------|-------|----|
| 1:0 | <p>PowerState - This 2-bit field is used both to determine the current power state of a Function and to set the Function into a new power state. The definition of the field values is given below.</p> <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>D3Hot</td> </tr> </table> <p>If software attempts to write an unsupported, optional state to this field, the write operation must complete normally; however, the data is discarded and no state change occurs.</p> <p>Default value of this field is 00b.</p> | 00b | D0 | 01b | D1 | 10b | D2 | 11b | D3Hot | RW |
| 00b | D0 | | | | | | | | | |
| 01b | D1 | | | | | | | | | |
| 10b | D2 | | | | | | | | | |
| 11b | D3Hot | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------------|
| 3 | <p>No_Soft_Reset - This bit indicates the state of the Function after writing the PowerState field to transition the Function from <u>D3_Hot</u> to <u>D0</u>. This bit <u>MUST</u>@FLIT be Set.</p> <p>When Set, this transition preserves internal Function state. The Function is in <u>D0_Active</u> and no additional software intervention is required.</p> <p>When Clear, this transition results in undefined internal Function state.</p> <p>Regardless of this bit, Functions that transition from <u>D3_Hot</u> to <u>D0</u> by Fundamental Reset must return to <u>D0_uninitialized</u> with only PME context preserved if PME is supported and enabled.</p> <p>If a VF implements the <u>PCI Power Management Capability</u>, the VF's value of this field must be identical to the associated PF's value.</p> | <u>RO</u> |
| 8 | <p>PME_En - When Set, the Function is permitted to generate a PME. When Clear, the Function is not permitted to generate a PME.</p> <p>If <u>PME_Support</u> is 1 xxxx b (PME generation from <u>D3_Cold</u>) or the Function consumes auxiliary power and auxiliary power is available this bit is <u>RWS</u> and the bit is not modified by Conventional Reset or <u>FLR</u>.</p> <p>If <u>PME_Support</u> is 0 xxxx b, this field is not sticky (<u>RW</u>) and defaults to 0b in response to a Conventional Reset or an <u>FLR</u>.</p> <p>If <u>PME_Support</u> is 0 0000 b, this bit is permitted to be hardwired to 0b.</p> | <u>RW / RWS</u> |
| 12:9 | <p>Data_Select - This 4-bit field is used to select which data is to be reported through the <u>Power Management Data Register</u> and <u>Data_Scale</u> field.</p> <p>If the <u>Power Management Data Register</u> is not implemented, this field must be hardwired to Zero.</p> <p>Refer to § <u>Section 7.5.2.3</u> for more details.</p> <p>The default of this field is Zero.</p> | <u>RW</u> <u>VF ROZ</u> |
| 14:13 | <p>Data_Scale - This field indicates the scaling factor to be used when interpreting the value of the <u>Data register</u>. The value and meaning of this field will vary depending on which data value has been selected by the <u>Data_Select</u> field.</p> <p>This field is a required component of the <u>Power Management Data Register</u> (offset 7) and must be implemented if the <u>Power Management Data Register</u> is implemented.</p> <p>If the <u>Power Management Data Register</u> is not implemented, this field must be hardwired to Zero.</p> <p>Refer to § <u>Section 7.5.2.3</u> for more details.</p> | <u>RO</u> <u>VF ROZ</u> |
| 15 | <p>PME_Status - This bit is Set when the Function would normally generate a PME signal. The value of this bit is not affected by the value of the <u>PME_En</u> bit.</p> <p>If <u>PME_Support</u> bit 31 of the <u>Power Management Capabilities Register</u> is Clear, this bit is permitted to be hardwired to 0b.</p> <p>Functions that consume auxiliary power must preserve the value of this sticky register when auxiliary power is available. In such Functions, this register value is not modified by Conventional Reset or <u>FLR</u>.</p> | <u>RW1CS</u> |
| 23:22 | Undefined Undefined - these bits were defined in previous specifications. They should be ignored by software. | <u>RO</u> |

7.5.2.3 Power Management Data Register (Offset 07h) §

The Power Management Data Register is an optional, 8-bit read-only register that provides a mechanism for the Function to report state dependent operating power consumed or dissipation.

If the Power Management Data Register is implemented, then the Data_Select and Data_Scale fields must also be implemented. If this register is not implemented, it must be hardwired to 00h.

Software may check for the presence of the Power Management Data Register by writing different values into the Data_Select field, looking for non-zero return data in the Power Management Data Register and/or Data_Scale field. Any non-zero Power Management Data Register / Data_Select read data indicates that the Power Management Data Register complex has been implemented.

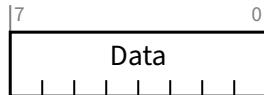


Figure 7-20 Power Management Data Register §

Table 7-16 Power Management Data Register §

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| 7:0 | Data - This register is used to report the state dependent data requested by the Data_Select field. The value of this register is scaled by the value reported by the Data_Scale field. For VFs, this register is not supported and must be hardwired to Zero. | RO VF ROZ |

The Power Management Data Register is used by writing the proper value to the Data_Select field in the PMCSR and then reading the Data_Scale field and the Power Management Data Register . The binary value read from Data is then multiplied by the scaling factor indicated by Data_Scale to arrive at the value for the desired measurement. § Table 7-17 shows which measurements are defined and how to interpret the values of each register.

Table 7-17 Power Consumption/Dissipation Reporting §

| Value in Data_Select | Data Reported | Data_Scale Interpretation | Units/Accuracy |
|----------------------|---|--|----------------|
| 0 | D0 Power Consumed | 0 = Unknown 1 = 0.1x 2 = 0.01x 3 = 0.001x | Watts |
| 1 | D1 Power Consumed | | |
| 2 | D2 Power Consumed | | |
| 3 | D3 Power Consumed | | |
| 4 | D0 Power Dissipated | | |
| 5 | D1 Power Dissipated | | |
| 6 | D2 Power Dissipated | | |
| 7 | D3 Power Dissipated | | |
| 8 | Common logic power consumption (Multi-Function Devices , Function 0 only) Function 0 of a Multi-Function Device : Power consumption that is not associated with a specific Function. All other Functions: Reserved | | |

| Value in Data_Select | Data Reported | Data_Scale Interpretation | Units/ Accuracy |
|----------------------|---------------|---------------------------|-----------------|
| 9-15 | Reserved | Reserved | TBD |

The “Power Consumed” values defined above must include all power consumed from the power planes through the connector pins. If the add-in card provides power to external devices, that power must be included as well. It must not include any power derived from a battery or an external source. This information is useful for management of the power supply or battery.

The “Power Dissipated” values must provide the amount of heat which will be released into the interior of the computer chassis. This excludes any power delivered to external devices but must include any power derived from a battery or external power source and dissipated inside the computer chassis. This information is useful for fine grained thermal management.

Multi-Function Devices are recommended to report the power consumed by each Function in each corresponding Function’s Configuration Space. In a Multi-Function Device , power consumption for circuitry common to multiple Functions is reported in Function 0’s Configuration Space through the Power Management Data Register once the Data_Select field of Function 0’s Power Management Control/Status Register has been programmed to 1000b. For a Multi-Function Device , power consumption of the device is the sum of this value and, for every Function of the device, the reported value associated with the Function’s current Power State.

Multiple component add-in cards implementing power reporting (i.e., multiple components behind a switch or bridge) must have the switch/bridge report the power it uses by itself. Each Function of each component on the add-in card is responsible for reporting the power consumed by that Function.

IMPLEMENTATION NOTE: NEW DESIGNS SHOULD USE POWER BUDGETING EXTENDED CAPABILITY §

Both the Power Budgeting Extended Capability and the PCI Power Management Capability report power consumption. The Power Budgeting Extended Capability mechanism is required in some situations (by this specification or the associated form factor specification). The Power Budgeting Extended Capability mechanism provides additional information beyond that which is provided by the Data register of the PCI Power Management Capability . It is strongly recommended that designs implement the Power Budgeting Extended Capability instead of the mechanism in this section, and that new designs hardwire the Data , Data_Select , and Data_Scale fields to 0.

7.5.3 PCI Express Capability Structure §

PCI Express defines a Capability structure in PCI-compatible Configuration Space (first 256 bytes) as shown in § Figure 7-3 . This structure allows identification of a PCI Express device Function and indicates support for new PCI Express features. The PCI Express Capability structure is required for PCI Express device Functions. The Capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express device Function, the PCI Express Capability structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

§ Figure 7-21 details allocation of register fields in the PCI Express Capability structure.

The PCI Express Capabilities, Device Capabilities , Device Status , and Device Control registers are required for all PCI Express device Functions. Device Capabilities 2 , Device Status 2 , and Device Control 2 registers are required for all PCI Express device Functions that implement capabilities requiring those registers. For device Functions that do not

implement the Device Capabilities 2 , Device Status 2 , and Device Control 2 registers, these spaces must be hardwired to 0b.

The Link Capabilities , Link Status , and Link Control registers are required for all Root Ports, Switch Ports, Bridges, and Endpoints that are not RCiEPs. For Functions that do not implement the Link Capabilities , Link Status , and Link Control registers, these spaces must be hardwired to 0. Link Capabilities 2 , Link Status 2 , and Link Control 2 registers are required for all Root Ports, Switch Ports, Bridges, and Endpoints (except for RCiEPs) that implement capabilities requiring those registers. For Functions that do not implement the Link Capabilities 2 , Link Status 2 , and Link Control 2 registers, these spaces must be hardwired to 0b.

The Slot Capabilities , Slot Status , and Slot Control registers are required in certain Switch Downstream and Root Ports. The Slot Capabilities Register is required if the Slot Implemented bit is Set (see § Section 7.5.3.2). The Slot Status and Slot Control registers are required if Slot Implemented is Set or if Data Link Layer Link Active Reporting Capable is Set (see § Section 7.5.3.6). Switch Downstream and Root Ports are permitted to implement these registers, even when they are not required, and in this case the behavior of most of the fields in these registers is undefined. See § Section 7.5.3.9 , § Section 7.5.3.10 , and § Section 7.5.3.11 for details. For Functions that do not implement the Slot Capabilities , Slot Status , and Slot Control registers, these spaces must be hardwired to 0b, with the exception of the Presence Detect State bit in the Slot Status Register of Downstream Ports, which must be hardwired to 1b (see § Section 7.5.3.11). Slot Capabilities 2 , Slot Status 2 , and Slot Control 2 registers are required for Switch Downstream and Root Ports if the Function implements capabilities requiring those registers. For Functions that do not implement the Slot Capabilities 2 , Slot Status 2 , and Slot Control 2 registers, these spaces must be hardwired to 0b.

Root Ports and Root Complex Event Collectors must implement the Root Capabilities , Root Status , and Root Control registers. For Functions that do not implement the Root Capabilities , Root Status , and Root Control registers, these spaces must be hardwired to 0b.

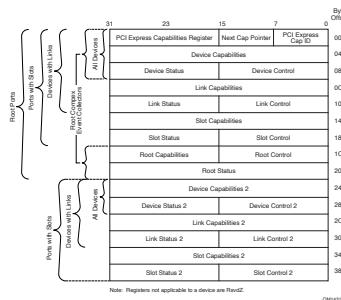


Figure 7-21 PCI Express Capability Structure §

7.5.3.1 PCI Express Capability List Register (Offset 00h) §

The PCI Express Capability List Register enumerates the PCI Express Capability structure in the PCI Configuration Space Capability list. § Figure 7-22 details allocation of register fields in the PCI Express Capability List Register ; § Table 7-18 provides the respective bit definitions.

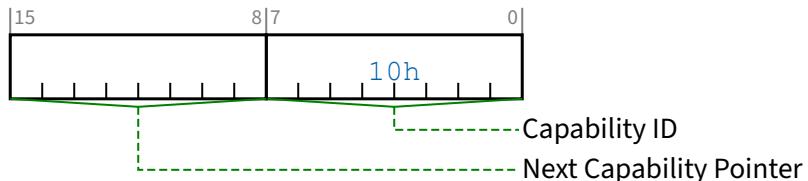


Figure 7-22 PCI Express Capability List Register §

Table 7-18 PCI Express Capability List Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Capability ID - Indicates the PCI Express Capability structure. This field must return a Capability ID of 10h indicating that this is a PCI Express Capability structure. | RO |
| 15:8 | Next Capability Pointer - This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities. | RO |

7.5.3.2 PCI Express Capabilities Register (Offset 02h) §

The PCI Express Capabilities Register identifies PCI Express device Function type and associated capabilities. § Figure 7-23 details allocation of register fields in the PCI Express Capabilities Register ; § Table 7-19 provides the respective bit definitions.

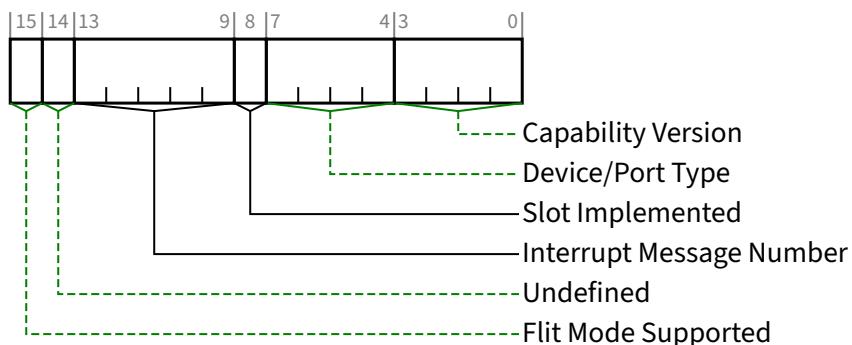


Figure 7-23 PCI Express Capabilities Register §

Table 7-19 PCI Express Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | Capability Version - Indicates PCI-SIG defined PCI Express Capability structure version number. A version of the specification that changes the PCI Express Capability structure in a way that is not otherwise identifiable (e.g., through a new Capability field) is permitted to increment this field. All such | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>changes to the PCI Express Capability structure must be software-compatible. Software must check for Capability Version numbers that are greater than or equal to the highest number defined when the software is written, as Functions reporting any such Capability Version numbers will contain a PCI Express Capability structure that is compatible with that piece of software.</p> <p>Must be hardwired to 2h for Functions compliant to this specification.</p> | |
| 7:4 | <p>Device/Port Type¹⁷⁸ - Indicates the specific type of this PCI Express Function. Note that different Functions in a Multi-Function Device can generally be of different types.</p> <p>Defined encodings for Functions that implement a Type 00h PCI Configuration Space header are:</p> <ul style="list-style-type: none"> 0000b PCI Express Endpoint 0001b Legacy PCI Express Endpoint 1001b RCIEP 1010b Root Complex Event Collector <p>Defined encodings for Functions that implement a Type 01h PCI Configuration Space header are:</p> <ul style="list-style-type: none"> 0100b Root Port of PCI Express Root Complex 0101b Upstream Port of PCI Express Switch 0110b Downstream Port of PCI Express Switch 0111b PCI Express to PCI/PCI-X Bridge 1000b PCI/PCI-X to PCI Express Bridge <p>All other encodings are Reserved.</p> <p>Note that the different Endpoint types have notably different requirements in § Section 1.3.2 regarding I/O resources, Extended Configuration Space, and other capabilities.</p> | RO |
| 8 | <p>Slot Implemented - When Set, this bit indicates that the Link associated with this Port is connected to a slot (as compared to being connected to a system-integrated device or being disabled).</p> <p>This bit is valid for Downstream Ports. This bit is undefined for Upstream Ports.</p> | HwInit |
| 13:9 | <p>Interrupt Message Number - When MSI/MSI-X is implemented, this field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this Capability structure. This vector is also used for:</p> <ul style="list-style-type: none"> Native PME Software Model (see § Section 6.1.6 and § Section 6.7.3.4) Link Activation (see § Section 5.5.6) Flit Error Counter Interrupts (see § Section 7.7.8.4) <p>For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control Register for MSI .</p> <p>For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.</p> | RO |

178. This field would be better named ‘Function Type’ but for historical reasons is named Device/Port Type.

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 14 | Undefined - The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate support for TCS Routing. System software should ignore the value read from this bit. System software is permitted to write any value to this bit. | RO |
| 15 | Flit Mode Supported - When Set, indicates support for Flit Mode. Must be Set by all implementations that support Flit Mode. Must be Clear by implementations that do not support Flit Mode. See Flit Mode Supported . | HwInit |

7.5.3.3 Device Capabilities Register (Offset 04h) §

The Device Capabilities Register identifies PCI Express device Function specific capabilities. § [Figure 7-24](#) details allocation of register fields in the Device Capabilities Register ; § [Table 7-20](#) provides the respective bit definitions.

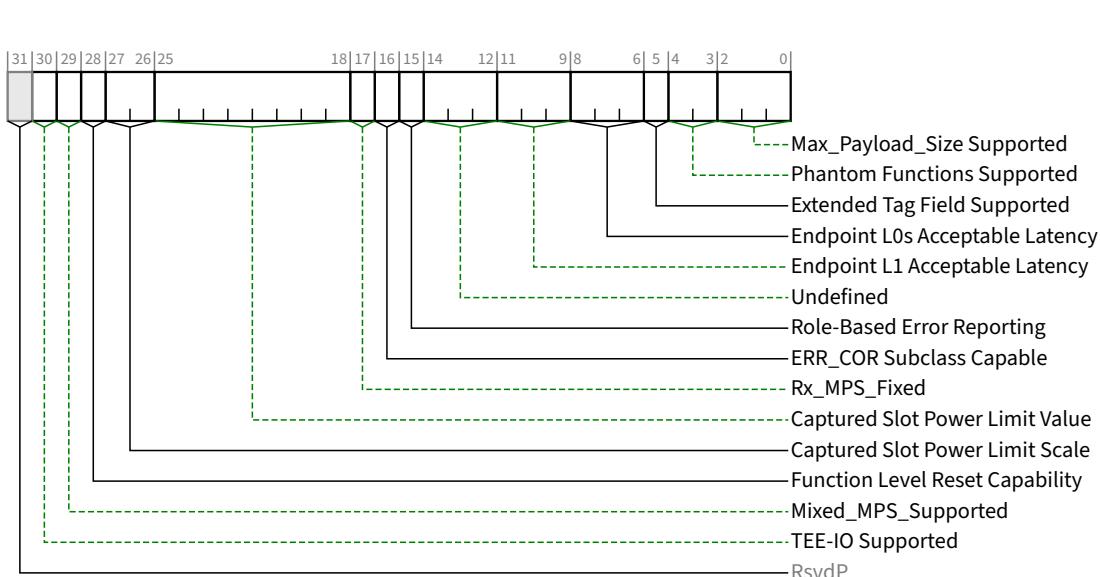


Figure 7-24 Device Capabilities Register §

Table 7-20 Device Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>Max_Payload_Size Supported - This field indicates the maximum payload size that the Function can support for TLPs. This field MUST @FLIT indicate a minimum of 512 bytes.</p> <p>If the Rx_MPS_Fixed bit is Set, the Function's Rx_MPS_Limit is fixed with the value indicated by this (Max_Payload_Size Supported) field. Otherwise, the Rx_MPS_Limit is determined by the Max_Payload_Size field (the "MPS setting") in one or more Functions. See § Section 2.2.2 for important details regarding Multi-Function Devices .</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b 128 bytes max payload size 001b 256 bytes max payload size 010b 512 bytes max payload size | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| | <p>01b 1024 bytes max payload size 100b 2048 bytes max payload size 101b 4096 bytes max payload size 110b Reserved 111b Reserved</p> <p>The Functions of a Multi-Function Device are permitted to report different values for this field.</p> | |
| 4:3 | <p>Phantom Functions Supported - This field indicates the support for use of unclaimed Function Numbers to extend the number of outstanding transactions allowed by logically combining unclaimed Function Numbers (called Phantom Functions) with the Tag identifier (see § Section 2.2.6.2 for a description of Tag Extensions).</p> <p>For a PF ↓↑of an SR-IOV Device with its ↓↑VF Enable ↓↑VF Enable bit Set, the use of Phantom Function numbers is not permitted and this field must return <u>Zero</u> when read.</p> <p>↓↑For a PF of an SIOV Device with its SIOV Enabled bit Set, the use of Phantom Function numbers is not permitted and this field must return Zero when read.↑ ECN: Base 6.3 SIOV△</p> <p>For VFs, this field is not supported and must be hardwired to <u>Zero</u>.</p> <p>For every Function in an <u>ARI Device</u>, this field must be hardwired to <u>Zero</u>.</p> <p>The remainder of this field description applies only to non-ARI Multi-Function Devices.</p> <p>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.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b No Function Number bits are used for Phantom Functions. Multi-Function Devices are permitted to implement up to 8 independent Functions. 01b The most significant bit of the Function number in Requester ID is used for Phantom Functions; a Multi-Function Device is permitted to implement Functions 0-3. Functions 0, 1, 2, and 3 are permitted to use Function Numbers 4, 5, 6, and 7 respectively as Phantom Functions. 10b The two most significant bits of Function Number in Requester ID are used for Phantom Functions; a Multi-Function Device is permitted to implement Functions 0-1. Function 0 is permitted to use Function Numbers 2, 4, and 6 for Phantom Functions. Function 1 is permitted to use Function Numbers 3, 5, and 7 as Phantom Functions. 11b All 3 bits of Function Number in Requester ID used for Phantom Functions. The device must have a single Function 0 that is permitted to use all other Function Numbers as Phantom Functions. <p>Note that Phantom Function support for the Function must be enabled by the <u>Phantom Functions Enable</u> field in the Device Control Register before the Function is permitted to use the Function Number field in the Requester ID for Phantom Functions.</p> | RO VF ROZ |
| 5 | <p>Extended Tag Field Supported - This bit, in combination with the 10-Bit Tag Requester Supported bit and the 14-Bit Tag Requester Supported bit, indicates the maximum supported size of the Tag field as a Requester. This bit must be Set if the ↓↑10-Bit Tag Requester Supported ↓↑10-Bit Tag Requester Supported bit or the ↓↑14-Bit Tag Requester Supported ↓↑14-Bit Tag Requester Supported bit is Set.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0b 5-bit Tag Requester capability supported 1b 8-bit Tag Requester capability supported | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|-------------|------------------|-------------|-------------------|-------------|-------------------|-------------|-------------------|-------------|------------------|-------------|------------------|-------------|------------------|-------------|----------|----|
| | Note that 8-bit Tag field generation must be enabled by the Extended Tag Field Enable bit in the Device Control Register of the Requester Function before 8-bit Tags can be generated by the Requester. See § Section 2.2.6.2 for interactions with enabling the use of 10-Bit or 14-Bit Tags. | | | | | | | | | | | | | | | | | |
| 8:6 | <p>Endpoint L0s Acceptable Latency - This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering.</p> <p>Power management software uses the reported L0s Acceptable Latency number to compare against the L0s exit latencies reported by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L0s entry can be used with no loss of performance.</p> <p>Defined encodings are:</p> <table> <tr><td>000b</td><td>Maximum of 64 ns</td></tr> <tr><td>001b</td><td>Maximum of 128 ns</td></tr> <tr><td>010b</td><td>Maximum of 256 ns</td></tr> <tr><td>011b</td><td>Maximum of 512 ns</td></tr> <tr><td>100b</td><td>Maximum of 1 µs</td></tr> <tr><td>101b</td><td>Maximum of 2 µs</td></tr> <tr><td>110b</td><td>Maximum of 4 µs</td></tr> <tr><td>111b</td><td>No limit</td></tr> </table> <p>For Functions other than Endpoints, this field is Reserved and must be hardwired to 000b.</p> | 000b | Maximum of 64 ns | 001b | Maximum of 128 ns | 010b | Maximum of 256 ns | 011b | Maximum of 512 ns | 100b | Maximum of 1 µs | 101b | Maximum of 2 µs | 110b | Maximum of 4 µs | 111b | No limit | RO |
| 000b | Maximum of 64 ns | | | | | | | | | | | | | | | | | |
| 001b | Maximum of 128 ns | | | | | | | | | | | | | | | | | |
| 010b | Maximum of 256 ns | | | | | | | | | | | | | | | | | |
| 011b | Maximum of 512 ns | | | | | | | | | | | | | | | | | |
| 100b | Maximum of 1 µs | | | | | | | | | | | | | | | | | |
| 101b | Maximum of 2 µs | | | | | | | | | | | | | | | | | |
| 110b | Maximum of 4 µs | | | | | | | | | | | | | | | | | |
| 111b | No limit | | | | | | | | | | | | | | | | | |
| 11:9 | <p>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.</p> <p>Power management software uses the reported L1 Acceptable Latency number to compare against the L1 Exit Latencies reported (see below) by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L1 entry can be used with no loss of performance.</p> <p>Defined encodings are:</p> <table> <tr><td>000b</td><td>Maximum of 1 µs</td></tr> <tr><td>001b</td><td>Maximum of 2 µs</td></tr> <tr><td>010b</td><td>Maximum of 4 µs</td></tr> <tr><td>011b</td><td>Maximum of 8 µs</td></tr> <tr><td>100b</td><td>Maximum of 16 µs</td></tr> <tr><td>101b</td><td>Maximum of 32 µs</td></tr> <tr><td>110b</td><td>Maximum of 64 µs</td></tr> <tr><td>111b</td><td>No limit</td></tr> </table> <p>For Functions other than Endpoints, this field is Reserved and must be hardwired to 000b.</p> | 000b | Maximum of 1 µs | 001b | Maximum of 2 µs | 010b | Maximum of 4 µs | 011b | Maximum of 8 µs | 100b | Maximum of 16 µs | 101b | Maximum of 32 µs | 110b | Maximum of 64 µs | 111b | No limit | RO |
| 000b | Maximum of 1 µs | | | | | | | | | | | | | | | | | |
| 001b | Maximum of 2 µs | | | | | | | | | | | | | | | | | |
| 010b | Maximum of 4 µs | | | | | | | | | | | | | | | | | |
| 011b | Maximum of 8 µs | | | | | | | | | | | | | | | | | |
| 100b | Maximum of 16 µs | | | | | | | | | | | | | | | | | |
| 101b | Maximum of 32 µs | | | | | | | | | | | | | | | | | |
| 110b | Maximum of 64 µs | | | | | | | | | | | | | | | | | |
| 111b | No limit | | | | | | | | | | | | | | | | | |
| 14:12 | Undefined - The value read from these bits are undefined. In previous versions of this specification, this bit was used to indicate that a Attention Button, Attention Indicator, or Power Indicator, is implemented on the adapter and electrically controlled by the component on the adapter. System software must ignore the value read from this bit. System software is permitted to write any value to this bit. | RO | | | | | | | | | | | | | | | | |
| 15 | Role-Based Error Reporting - When Set, this bit indicates that the Function implements Role-Based Error Reporting . This bit must be Set by all Functions conforming to [PCIe-1.1] or later revisions. | RO | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|---|------------|------|------------|------|------------|-------|------------|--------|----|
| 16 | <p>ERR_COR Subclass Capable - When Set, this bit indicates that the Function supports the ERR_COR Subclass field in ERR_COR Messages, allowing different subclasses to be distinguished. See § Section 2.2.8.3.</p> <p>Downstream Ports that implement the System Firmware Intermediary (SFI) capability must Set this bit. Downstream Ports that implement Downstream Port Containment (DPC) are strongly encouraged to Set this bit.</p> | RO | | | | | | | | |
| 17 | <p>Rx MPS Fixed – When Set, the Function's Rx_MPS_Limit is fixed with the value indicated by the Max_Payload_Size Supported field. Otherwise, the Rx_MPS_Limit is determined by the Max_Payload_Size field (the "MPS setting") in one or more Functions. See § Section 2.2.2 for important details regarding Multi-Function Devices . This bit <i>MUST@FLIT</i> be Set.</p> | HwInit | | | | | | | | |
| 25:18 | <p>Captured Slot Power Limit Value (Upstream Ports only) - In combination with the Captured Slot Power Limit Scale value, specifies the upper limit on power available to the adapter.</p> <p>Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Captured Slot Power Limit Scale field except when the Captured Slot Power Limit Scale field equals 00b (1.0x) and the Captured Slot Power Limit Value exceeds EFh, then alternative encodings are used (see § Section 7.5.3.9).</p> <p>This value is set by the Set_Slot_Power_Limit Message or hardwired to 00h (see § Section 6.9). The default value is 00h.</p> <p>For VFs, the field value when read is undefined.</p> | RO | | | | | | | | |
| 27:26 | <p>Captured Slot Power Limit Scale (Upstream Ports only) - Specifies the scale used for the Slot Power Limit Value .</p> <p>Range of Values:</p> <table> <tr> <td>00b</td> <td>1.0x</td> </tr> <tr> <td>01b</td> <td>0.1x</td> </tr> <tr> <td>10b</td> <td>0.01x</td> </tr> <tr> <td>11b</td> <td>0.001x</td> </tr> </table> <p>This value is set by the Set_Slot_Power_Limit Message or hardwired to 00b (see § Section 6.9). The default value is 00b.</p> <p>For VFs, the field value when read is undefined.</p> | 00b | 1.0x | 01b | 0.1x | 10b | 0.01x | 11b | 0.001x | RO |
| 00b | 1.0x | | | | | | | | | |
| 01b | 0.1x | | | | | | | | | |
| 10b | 0.01x | | | | | | | | | |
| 11b | 0.001x | | | | | | | | | |
| 28 | <p>Function Level Reset Capability - A value of 1b indicates the Function supports the optional Function Level Reset mechanism described in § Section 6.6.2 .</p> <p>This bit applies to Endpoints only. For all other Function types this bit must be hardwired to <u>Zero</u> .</p> <p>For PFs and VFs, the feature is mandatory and this bit must be Set.</p> | RO | | | | | | | | |
| 29 | <p>Mixed MPS Supported – When Set, the Function must have an implementation specific mechanism capable of supporting different MPS settings for different targets. This bit <i>MUST@FLIT</i> be Set if the Function supports P2P Memory Transactions and the Function's Max_Payload_Size Supported field indicates an MPS value greater than 512 bytes. If not mandatory, supporting Mixed MPS capability may still be beneficial if this Function does P2P with targets or over paths whose supported MPS is significantly less than this Function's supported MPS; e.g., 128 bytes vs. 512 bytes.</p> <p>The implementation specific mechanism must handle both Request and Completion TLPs, and is permitted to base its determination of P2P targets on Memory Space ranges, Bus Number ranges, or implementation specific means; e.g., data mover channels.</p> <p>For N SR-IOV devices, N SR-IOV Devices, this field in each VF must have the same value its associated PF. If this field is Set, the implementation specific mechanism must use the same P2P target-specific MPS setting for each VF as its associated PF. This parallels the requirement for the Max_Payload_Size field in the Device Control Register .</p> | HwInit | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 30 | TEE-IO Supported – When Set, this bit indicates that the Function implements the TEE-IO functionality as described by the TEE Device Interface Security Protocol (TDISP). See § Chapter 11.. | HwInit |

7.5.3.4 Device Control Register (Offset 08h) §

The Device Control Register controls PCI Express device specific parameters. § Figure 7-25 details allocation of register fields in the Device Control Register ; § Table 7-21 provides the respective bit definitions.

For VF fields indicated as RsvdP , the PF setting applies to the VF.

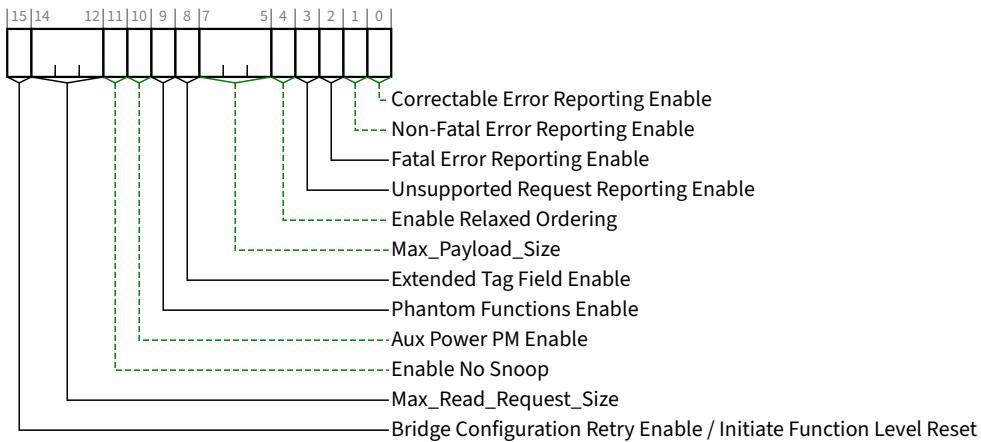


Figure 7-25 Device Control Register §

Table 7-21 Device Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 0 | <p>Correctable Error Reporting Enable - This bit, in conjunction with other bits, controls sending ERR_COR Messages (see § Section 6.2.5 , § Section 6.2.6 , and § Section 6.2.11.2 for details). For a Multi-Function Device , this bit controls error reporting for each Function from point-of-view of the respective Function.</p> <p>For a Root Port, the reporting of correctable errors is internal to the root. No external ERR_COR Message is generated.</p> <p>An RCIEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP |
| 1 | <p>Non-Fatal Error Reporting Enable - This bit, in conjunction with other bits, controls sending ERR_NONFATAL Messages (see § Section 6.2.5 and § Section 6.2.6 for details). For a Multi-Function Device , this bit controls error reporting for each Function from point-of-view of the respective Function.</p> <p>For a Root Port, the reporting of Non-fatal errors is internal to the root. No external ERR_NONFATAL Message is generated.</p> <p>An RCIEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|----------------|---------------|-------------|---------------|-------------|---------------|-------------|----------------|-------------|----------------|-------------|----------------|-------------|----------|-------------|----------|----------------|
| 2 | <p>Fatal Error Reporting Enable - This bit, in conjunction with other bits, controls sending ERR_FATAL Messages (see § Section 6.2.5 and § Section 6.2.6 for details). For a Multi-Function Device , this bit controls error reporting for each Function from point-of-view of the respective Function.</p> <p>For a Root Port, the reporting of Fatal errors is internal to the root. No external ERR_FATAL Message is generated.</p> <p>An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP | | | | | | | | | | | | | | | | |
| 3 | <p>Unsupported Request Reporting Enable - This bit, in conjunction with other bits, controls the signaling of Unsupported Request Errors by sending error Messages (see § Section 6.2.5 and § Section 6.2.6 for details). For a Multi-Function Device , this bit controls error reporting for each Function from point-of-view of the respective Function.</p> <p>An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP | | | | | | | | | | | | | | | | |
| 4 | <p>Enable Relaxed Ordering - If this bit is Set, the Function is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering (see § Section 2.2.6.4 and § Section 2.4).</p> <p>A Function is permitted to hardwire this bit to 0b if it never sets the Relaxed Ordering attribute in transactions it initiates as a Requester.</p> <p>When not hardwired to 0b, the default value of this bit is 1b.</p> | RW VF RsvdP | | | | | | | | | | | | | | | | |
| 7:5 | <p>Max_Payload_Size - For specified cases, this field determines the maximum TLP payload size (the MPS setting) for the Function. Values permitted to be programmed are indicated by the Max_Payload_Size Supported field.</p> <p>As a Receiver, if the Rx_MPS_Fixed bit is Set, the Rx_MPS_Limit is fixed with the value indicated by the Max_Payload_Size Supported field. Otherwise, the Rx_MPS_Limit is determined by the MPS setting in one or more Functions. See § Section 2.2.2 for important details regarding Multi-Function Devices .</p> <p>As a Transmitter, the Function must not generate TLPs with payloads exceeding the MPS setting, with the exception of Functions in a Multi-Function Device , or Functions with implementation-specific mechanisms capable of supporting different MPS settings for different targets. See § Section 2.2.2 for important details.</p> <p>Defined encodings for this field are:</p> <table style="margin-left: 20px;"> <tr> <td>000b</td> <td>128 bytes MPS</td> </tr> <tr> <td>001b</td> <td>256 bytes MPS</td> </tr> <tr> <td>010b</td> <td>512 bytes MPS</td> </tr> <tr> <td>011b</td> <td>1024 bytes MPS</td> </tr> <tr> <td>100b</td> <td>2048 bytes MPS</td> </tr> <tr> <td>101b</td> <td>4096 bytes MPS</td> </tr> <tr> <td>110b</td> <td>Reserved</td> </tr> <tr> <td>111b</td> <td>Reserved</td> </tr> </table> <p>Functions that support only the 128-byte MPS are permitted to hardwire this field to 000b.</p> <p>System software is not required to program the same value for this field for all the Functions of a Multi-Function Device .</p> <p>Default value of this field is 000b.</p> | 000b | 128 bytes MPS | 001b | 256 bytes MPS | 010b | 512 bytes MPS | 011b | 1024 bytes MPS | 100b | 2048 bytes MPS | 101b | 4096 bytes MPS | 110b | Reserved | 111b | Reserved | RW VF RsvdP |
| 000b | 128 bytes MPS | | | | | | | | | | | | | | | | | |
| 001b | 256 bytes MPS | | | | | | | | | | | | | | | | | |
| 010b | 512 bytes MPS | | | | | | | | | | | | | | | | | |
| 011b | 1024 bytes MPS | | | | | | | | | | | | | | | | | |
| 100b | 2048 bytes MPS | | | | | | | | | | | | | | | | | |
| 101b | 4096 bytes MPS | | | | | | | | | | | | | | | | | |
| 110b | Reserved | | | | | | | | | | | | | | | | | |
| 111b | Reserved | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|---|
| 8 | <p>Extended Tag Field Enable - This bit, in combination with the 10-Bit Tag Requester Enable bit and the 14-Bit Tag Requester Enable bit, determines how many Tag field bits a Requester is permitted to use for non-UIO Requests. ↑↓This bit is not applicable to a Requester when generating UIO Requests.↑</p> <p>The following applies when the 10-Bit Tag Requester Enable bit and the 14-Bit Tag Requester Enable bit are both Clear. If the Extended Tag Field Enable bit is Set, the Function is permitted to use an 8-bit Tag field as a Requester. If the bit is Clear, the Function is restricted to using a 5-bit Tag field.</p> <p>See § Section 2.2.6.2 for required behavior when one or both of these ↑↓larger Tag Requester Enable bits↓↑↓larger Tag Requester Enable bits↑ are Set.</p> <p>If software changes the value of the Extended Tag Field Enable bit while the Function has outstanding Non-Posted Requests, the result is undefined.</p> <p>Functions that do not implement this capability hardwire this bit to 0b.</p> <p>Default value of this bit is implementation specific.</p> | Errata: Base 6.3 B807△↔ |
| 9 | <p>Phantom Functions Enable - This ↑↓bit, in combination with the 10-Bit Tag Requester Enable↓bit ↑↓and the 14-Bit Tag Requester Enable bit,↓ determines how many outstanding Non-Posted Requests a Requester is permitted to generate. ↑↓Non non-UIO Requests, the Extended Tag Field Enable , 10-Bit Tag Requester Enable , and 14-Bit Tag Requester Enable bits also apply.↑ See § Section 2.2.6.2 for complete details.</p> <p>When Set, this bit enables a Function to use unclaimed Functions as Phantom Functions to extend the number of outstanding transaction identifiers. If the bit is Clear, the Function is not allowed to use Phantom Functions.</p> <p>Behavior is undefined when this bit is Set in Functions with enabled Shadow Functions.</p> <p>Software should not change the value of this bit while the Function has outstanding Non-Posted Requests; otherwise, the result is undefined.</p> <p>Functions that do not implement this capability hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | Errata: Base 6.3 B807△↔ |
| 10 | <p>Aux Power PM Enable - When Set this bit, enables a Function to draw auxiliary power independent of PME Aux power. Functions that require auxiliary power on legacy operating systems should continue to indicate PME Aux power requirements. Auxiliary power is allocated as requested in the Aux_Current field of the Power Management Capabilities Register (PMC), independent of the PME_En bit in the Power Management Control/Status Register (PMCSR) (see § Chapter 5.). For Multi-Function Devices , a component is allowed to draw auxiliary power if at least one of the Functions has this bit set.</p> <p>Note: Functions that consume auxiliary power must preserve the value of this sticky register when auxiliary power is available. In such Functions, this bit is not modified by Conventional Reset.</p> <p>Functions that do not implement this capability hardwire this bit to 0b.</p> <p>Additional Aux power is permitted to be allocated using the firmware based mechanism (see the Request D3 Cold Aux Power Limit _DSM call as defined in [Firmware]).</p> <p>Additional Aux power is also permitted to be allocated by selecting a PM Sub State in the Power Limit mechanism (see § Section 7.8.1.3).</p> | RWS VF RsvdP |
| 11 | <p>Enable No Snoop - If this bit is Set, the Function is permitted to Set the No Snoop bit in the Requester Attributes of transactions it initiates that do not require hardware enforced cache coherency (see § Section 2.2.6.5). Note that setting this bit to 1b should not cause a Function to Set the No Snoop attribute on all transactions that it initiates. Even when this bit is Set, a Function is only permitted to Set the No Snoop attribute on a transaction when it can guarantee that the address of the transaction is not stored in any cache in the system.</p> <p>This bit is permitted to be hardwired to 0b if a Function would never Set the No Snoop attribute in transactions it initiates.</p> <p>Default value of this bit is 1b.</p> | RW VF RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|--|---|-------------------------------------|-------------|-------------------------------------|-------------|-------------------------------------|-------------|--------------------------------------|-------------|--------------------------------------|-------------|--------------------------------------|-------------|----------|-------------|----------|------------------------------|
| 14:12 | <p>Max_Read_Request_Size - This field sets the maximum Read Request size for the Function as a Requester. The Function must not generate Read Requests with a size exceeding the set value. Defined encodings for this field are:</p> <table> <tr><td>000b</td><td>128 bytes maximum Read Request size</td></tr> <tr><td>001b</td><td>256 bytes maximum Read Request size</td></tr> <tr><td>010b</td><td>512 bytes maximum Read Request size</td></tr> <tr><td>011b</td><td>1024 bytes maximum Read Request size</td></tr> <tr><td>100b</td><td>2048 bytes maximum Read Request size</td></tr> <tr><td>101b</td><td>4096 bytes maximum Read Request size</td></tr> <tr><td>110b</td><td>Reserved</td></tr> <tr><td>111b</td><td>Reserved</td></tr> </table> <p>Functions that do not generate Read Requests larger than 128 bytes and Functions that do not generate Read Requests on their own behalf are permitted to implement this field as Read Only (RO) with a value of 000b.</p> <p>Default value of this field is 010b.</p> | 000b | 128 bytes maximum Read Request size | 001b | 256 bytes maximum Read Request size | 010b | 512 bytes maximum Read Request size | 011b | 1024 bytes maximum Read Request size | 100b | 2048 bytes maximum Read Request size | 101b | 4096 bytes maximum Read Request size | 110b | Reserved | 111b | Reserved | <u>RW</u> <u>VF RsvdP</u> |
| 000b | 128 bytes maximum Read Request size | | | | | | | | | | | | | | | | | |
| 001b | 256 bytes maximum Read Request size | | | | | | | | | | | | | | | | | |
| 010b | 512 bytes maximum Read Request size | | | | | | | | | | | | | | | | | |
| 011b | 1024 bytes maximum Read Request size | | | | | | | | | | | | | | | | | |
| 100b | 2048 bytes maximum Read Request size | | | | | | | | | | | | | | | | | |
| 101b | 4096 bytes maximum Read Request size | | | | | | | | | | | | | | | | | |
| 110b | Reserved | | | | | | | | | | | | | | | | | |
| 111b | Reserved | | | | | | | | | | | | | | | | | |
| 15 | <p>Bridge Configuration Retry Enable / Initiate Function Level Reset - this bit has a different meaning based on Function type:</p> <ul style="list-style-type: none"> • PCI Express to PCI/PCI-X Bridges: <p>Bridge Configuration Retry Enable - When Set, this bit enables PCI Express to PCI/PCI-X bridges to return Request Retry Status (RRS) in response to Configuration Requests that target devices below the bridge. Refer to [PCIe-to-PCI-PCI-X-Bridge] for further details.</p> <p>Default value of this bit is 0b.</p> • Endpoints with Function Level Reset Capability set to 1b: <p>Initiate Function Level Reset - A write of 1b initiates Function Level Reset to the Function. The value read by software from this bit is always 0b.</p> <p>PFs and VFs must support FLR . Note: ↑↑performing↓↓Performing↑↑ ECN: Base 6.3 SIOV△↔ FLR on a PF ↑↑with an SR-IOV Extended Capability Clears↑↑ Clears its ↑↑VF Enabled↓↓VF Enabled↑↑ bit, which causes its VFs no longer to exist after the FLR completes. ↑↑Performing FLR on a PF with a Scalable IOV Extended Capability clears its SDI Enabled bit, which causes its SDIs to be disabled after the FLR completes.↑↑</p> • All others: <p>Reserved - Must hardwire the bit to 0b.</p> | <u>PCI Express to PCI/PCI-X Bridges:</u> <u>RW</u> <u>FLR</u> <u>Capable Endpoints:</u> <u>RW</u> <u>All others:</u> <u>RsvdP</u> | | | | | | | | | | | | | | | | |

IMPLEMENTATION NOTE: SOFTWARE UR REPORTING COMPATIBILITY WITH 1.0A DEVICES §

With [PCIe-1.0a] device Functions,¹⁷⁹ if the Unsupported Request Reporting Enable bit is Set, the Function when operating as a Completer will send an uncorrectable error Message (if enabled) when a UR error is detected. On platforms where an uncorrectable error Message is handled as a System Error, this will break PC-compatible Configuration Space probing, so software/firmware on such platforms may need to avoid setting the Unsupported Request Reporting Enable bit.

With device Functions implementing Role-Based Error Reporting, setting the Unsupported Request Reporting Enable bit will not interfere with PC-compatible Configuration Space probing, assuming that the severity for UR is left at its default of non-fatal. However, setting the Unsupported Request Reporting Enable bit will enable the Function to report UR errors¹⁸⁰ detected with posted Requests, helping avoid this case for potential silent data corruption.

On platforms where robust error handling and PC-compatible Configuration Space probing is required, it is suggested that software or firmware have the Unsupported Request Reporting Enable bit Set for Role-Based Error Reporting Functions, but clear for [PCIe-1.0a] Functions. Software or firmware can distinguish the two classes of Functions by examining the Role-Based Error Reporting bit in the Device Capabilities Register.

179. In this context, [PCIe-1.0a] devices Functions are devices that do not implement Role-Based Error Reporting .

180. With Role-Based Error Reporting devices, setting the SERR# Enable bit in the Command Register also implicitly enables UR reporting.

IMPLEMENTATION NOTE: USE OF MAX_PAYLOAD_SIZE §

The Max_Payload_Size (MPS) mechanism enables software to control the maximum payload in TLPs sent by Endpoints to balance latency versus bandwidth trade-offs, particularly for isochronous traffic.

If software chooses to program the MPS of various System Elements to non-default values, it must take care to ensure that each TLP with a data payload does not exceed the MPS setting of any System Element along the TLP's path. Otherwise, the TLP will be rejected by the System Element whose MPS setting is too small.

No specific algorithm to configure MPS is required by this specification, but software should base its algorithm upon factors such as the following:

- the MPS capability of each System Element within a Hierarchy
- awareness of when System Elements are added or removed through Hot-Plug operations
- knowing which System Elements send TLPs to each other, what type of traffic is carried, what type of transactions are used, and if TLP sizes are constrained by other mechanisms

For the case of system firmware that configures System Elements in preparation for running legacy operating system environments, system firmware may need to avoid programming MPS settings above the default of 128 bytes, which is the minimum supported by Endpoints.

For example, if the operating system environment does not implement services for optimizing MPS settings, system firmware probably should not program a non-default MPS for a Hierarchy that supports Hot-Plug operations. Otherwise, if no software is present to manage MPS settings when a new element is added, improper operation may result. Note that a newly added element may not even support a MPS setting as large as the rest of the Hierarchy, in which case software may need to deny enabling the new element or reduce the MPS settings of other elements, which may require quiescing all traffic carrying data payloads .

For ARI Devices and other MFDs, it's challenging to describe concisely what determines a Function's MPS limit for received TLPs. For this reason, the formal term Rx_MPS_Limit was introduced. It is used in many instances where former revisions of this specification used Max_Payload_Size in the context of a Receiver. It covers several special cases where the MPS limit is determined by the MPS settings in other Functions of an MFD. See § Section 2.2.2 for details.

For ARI Devices and other MFDs, it's also challenging to describe concisely what determines a Function's MPS limit for *transmitted* TLPs. For this reason, the formal term Tx_MPS_Limit was introduced. It is used in several instances where former revisions of this specification used Max_Payload_Size in the context of a Transmitter. It covers several special cases where the MPS limit is determined by the MPS settings in other Functions of an MFD. See § Section 2.2.2 for details.

IMPLEMENTATION NOTE: RX MPS FIXED ENHANCEMENT FOR MAX PAYLOAD SIZE §

The Rx_MPS_Fixed field was added to the Device Capabilities Register in the 6.0 Revision of this specification. As required in § Section 7.5.3.3, the Rx_MPS_Fixed capability bit *MUST* be Set.

When Rx_MPS_Fixed is Set, the Receiver MPS limit for that Function is the value of the Function's Max_Payload_Size Supported capability field, the highest MPS setting that the Function supports. The Rx_MPS_Fixed mechanism enables the MPS limit for the Function's Receiver and Transmitter to be independent, which in certain cases enables software to change MPS settings without having to quiesce all traffic carrying data payloads.

For example, in configurations where active Functions lack this enhancement, if software increases the MPS setting of a given Function, any TLPs with data payloads that the Function sends to another Function may exceed its MPS setting, resulting in Malformed TLP errors. ~~↑↓Similarly, ↓↑↑Similarly, ↑~~ if software decreases the MPS setting of a given Function, any TLPs with data payloads that other Functions send to it may exceed its MPS setting, again resulting in Malformed TLP errors. Without Rx_MPS_Fixed, the only general solution is to quiesce said traffic during reconfiguration.

IMPLEMENTATION NOTE: MIXED MAX PAYLOAD SIZE CONFIGURATIONS §

The simplest way for System Software to configure a non-default Max_Payload_Size (MPS) setting for a Hierarchy is to scan all Functions, determine the smallest Max_Payload_Size Supported capability, and configure the MPS setting in all Functions to this value. This guarantees that no Function will send a TLP with a payload size that the target Function can't handle. However, this simple policy may be unnecessarily restrictive, given that not all Functions send each other Memory Space transactions. In fact, many Endpoints only exchange Memory Space transactions with the host, and don't exchange any P2P TLPs with other Endpoints.

To support the use case for "mixed MPS configurations", a Function that has its Mixed_MPS_Supported bit Set is permitted to transmit TLPs with payloads exceeding its MPS setting, though it must never exceed its Max_Payload_Size Supported capability. For the case where host memory supports Rx_MPS_Fixed, System Software may configure the MPS setting in each Endpoint based solely on its path to host memory and the Max_Payload_Size Supported capability of host memory. Then, for any Endpoints that support P2P with other Endpoints, driver software can make adjustments necessary for the P2P traffic, including routing element MPS capability along P2P paths. If the Endpoint's Mixed_MPS_Supported bit is Set, indicating that it supports an implementation specific mechanism capable of supporting different MPS settings for different targets, the driver software may configure that mechanism to optimize the MPS setting for P2P target Endpoints. If the Endpoint does not support this type of mechanism, or if the mechanism is unable to accommodate all of the Endpoint's P2P MPS requirements, the driver software may reduce its MPS setting if needed to accommodate its P2P traffic.

Mixed MPS configurations are especially useful for cases where a set of Endpoints exchange a high volume of very large P2P TLPs between each other; e.g., a set of high-end accelerators or SSDs connected by one or more high-end switches. Such configurations might use a much larger MPS setting (e.g., 2048 bytes) for the high-end switches and accelerators/SSDs than supported by most hosts (e.g., 512 bytes).

IMPLEMENTATION NOTE: USE OF MAX_READ_REQUEST_SIZE §

The Max_Read_Request_Size mechanism allows improved control of bandwidth allocation in systems where Quality of Service (QoS) is important for the target applications. For example, an arbitration scheme based on counting Requests (and not the sizes of those Requests) provides imprecise bandwidth allocation when some Requesters use much larger sizes than others. The Max_Read_Request_Size mechanism can be used to force more uniform allocation of bandwidth, by restricting the upper size of Read Requests.

7.5.3.5 Device Status Register (Offset 0Ah) §

The Device Status Register provides information about PCI Express device (Function) specific parameters. § Figure 7-26 details allocation of register fields in the Device Status Register ; § Table 7-22 provides the respective bit definitions.

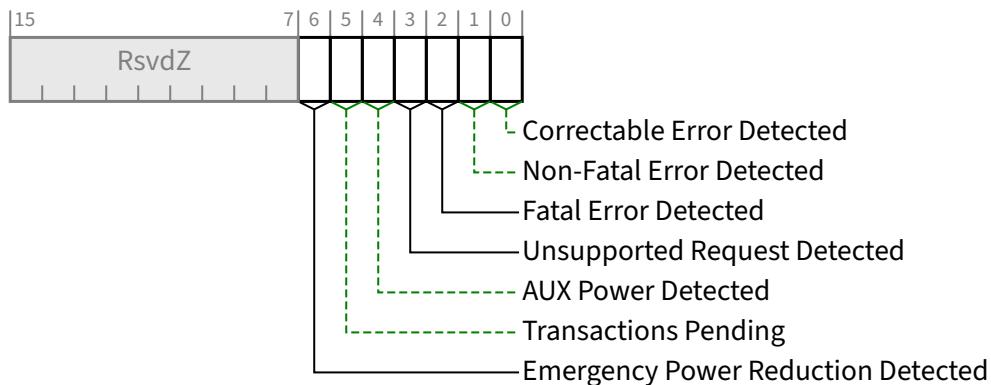


Figure 7-26 Device Status Register §

Table 7-22 Device Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>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 <u>Device Control Register</u>. For a Multi-Function Device , each Function indicates status of errors as perceived by the respective Function.</p> <p>For Functions supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the Correctable Error Mask register.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 1 | <p>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 <u>Device Control Register</u>. For a Multi-Function Device , each Function indicates status of errors as perceived by the respective Function.</p> <p>For Functions supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the Uncorrectable Error Mask register.</p> <p>Default value of this bit is 0b.</p> | RW1C |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| 2 | <p>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.</p> <p>For Functions supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the Uncorrectable Error Mask register.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 3 | <p>Unsupported Request Detected - This bit indicates that the Function received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register . For a Multi-Function Device , each Function indicates status of errors as perceived by the respective Function.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 4 | <p>AUX Power Detected - Functions that require auxiliary power report this bit as Set if auxiliary power is detected by the Function.</p> <p>For VFs, this bit is not supported and must be hardwired to Zero .</p> | RO VF ROZ |
| 5 | <p>Transactions Pending -</p> <p><i>Endpoints:</i></p> <p>When Set, this bit indicates that the Function ↓↓has issued Non-Posted Requests that have not been completed.↓↓is expecting one or more Completions.↑↑ A Function reports this bit cleared only ↓↓when all outstanding↓↓the Function is not expecting any Completions (All issued↓↓ Non-Posted Requests ↓↓and UIO Requests↓↓ have completed or have been terminated by the Completion Timeout mechanism. This ↓↓includes Requests that are issued by a host when taking ownership of a peer-to-peer Transaction.) This bit must also be cleared upon the completion of an FLR.</p> <p><i>Root and Switch Ports:</i></p> <p>When Set, this bit indicates that a Port has issued Non-Posted Requests ↓↓or UIO Requests↓↓ on its own behalf (using the Port's own, or its Shadow Function's, Requester ID) which have not been completed. The Port reports this bit cleared only when all such outstanding Non-Posted Requests ↓↓or UIO Requests↓↓ have completed or have been terminated by the Completion Timeout mechanism. Note that Root and Switch Ports implementing only the functionality required by this document do not issue Non-Posted Requests on their own behalf, and therefore are not subject to this case. Root and Switch Ports that do not issue Non-Posted Requests on their own behalf hardwire this bit to 0b.</p> | RO |
| 6 | <p>Emergency Power Reduction Detected - This bit is Set when the Function is in the Emergency Power Reduction State . Whenever any condition is present that would cause the Emergency Power Reduction State to be entered, the Function remains in the Emergency Power Reduction State and writes to this bit have no effect. See § Section 6.24 for additional details.</p> <p>Multi-Function Devices associated with an Upstream Port must Set this bit in all Functions that support Emergency Power Reduction State .</p> <p>For VFs, this bit is not supported and must be hardwired to Zero .</p> <p>Except for VFs, this bit is RsvdZ if the Emergency Power Reduction Supported field is 00b (see § Section 7.5.3.15).</p> <p>This bit is RsvdZ in Functions that are not associated with an Upstream Port.</p> <p>Default value is 0b.</p> | RW1C VF ROZ |

7.5.3.6 Link Capabilities Register (Offset 0Ch) §

The Link Capabilities Register identifies PCI Express Link specific capabilities. § Figure 7-27 details allocation of register fields in the Link Capabilities Register ; § Table 7-23 provides the respective bit definitions.

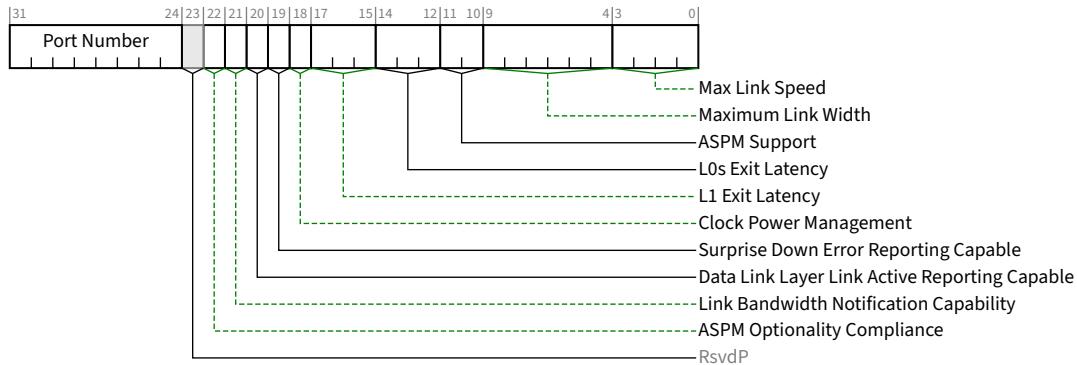


Figure 7-27 Link Capabilities Register §

Table 7-23 Link Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | <p>Max Link Speed - This field indicates the maximum Link speed of the associated Port. The encoded value specifies a Bit Location in the Supported Link Speeds Vector (in the Link Capabilities 2 Register) that corresponds to the maximum Link speed.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 <p>All other encodings are reserved.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | RO |
| 9:4 | <p>Maximum Link Width - This field indicates the maximum Link width (xN - corresponding to N Lanes) implemented by the component. This value is permitted to exceed the number of Lanes routed to the slot (Downstream Port), adapter connector (Upstream Port), or in the case of component-to-component connections, the actual wired connection width.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00 0001b x1 00 0010b x2 00 0100b x4 | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>00 1000b x8 01 0000b x16</p> <p>All other encodings are Reserved.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | |
| 11:10 | <p>ASPM Support / Active State Power Management Support - This field indicates the level of ASPM supported on the given PCI Express Link. See § Section 5.4.1 for ASPM support requirements.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b No ASPM Support 01b L0s Supported 10b L1 Supported 11b L0s and L1 Supported <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | RO |
| 14:12 | <p>L0s Exit Latency - This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s to L0. If L0s is not supported, the value is undefined; however, see the Implementation Note “Potential Issues With Legacy Software When L0s is Not Supported” in § Section 5.4.1.1 for the recommended value.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b Less than 64 ns 001b 64 ns to less than 128 ns 010b 128 ns to less than 256 ns 011b 256 ns to less than 512 ns 100b 512 ns to less than 1 µs 101b 1 µs to less than 2 µs 110b 2 µs-4 µs 111b More than 4 µs <p>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.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | RO |
| 17:15 | <p>L1 Exit Latency - This field indicates the L1 Exit Latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from ASPM L1 to L0. If ASPM L1 is not supported, the value is undefined.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b Less than 1 µs 001b 1 µs to less than 2 µs 010b 2 µs to less than 4 µs 011b 4 µs to less than 8 µs 100b 8 µs to less than 16 µs 101b 16 µs to less than 32 µs 110b 32 µs-64 µs | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>111b More than 64 µs</p> <p>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.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | |
| 18 | <p>Clock Power Management - For Upstream Ports, a value of 1b in this bit indicates that the component tolerates the removal of any reference clock(s) via the “clock request” (CLKREQ#) mechanism when the Link is in the L1 and L2/L3 Ready Link states. A value of 0b indicates the component does not have this capability and that reference clock(s) must not be removed in these Link states.</p> <p>L1 PM Substates defines other semantics for the CLKREQ# signal, which are managed independently of Clock Power Management .</p> <p>This Capability is applicable only in form factors that support “clock request” (CLKREQ#) capability.</p> <p>For a Multi-Function Device associated with an Upstream Port, each Function indicates its capability independently. Power Management configuration software must only permit reference clock removal if all Functions of the Multi-Function Device indicate a 1b in this bit. For ARI Devices , all Functions must indicate the same value in this bit.</p> <p>For Downstream Ports, this bit must be hardwired to 0b.</p> | RO |
| 19 | <p>Surprise Down Error Reporting Capable - For a Downstream Port, this bit must be Set if the component supports the optional capability of detecting and reporting a Surprise Down error condition.</p> <p>For Upstream Ports and components that do not support this optional capability, this bit must be hardwired to 0b.</p> | RO |
| 20 | <p>Data Link Layer Link Active Reporting Capable - For a Downstream Port, this bit must be hardwired to 1b if the component supports the optional capability of reporting the DL_Active state of the Data Link Control and Management State Machine. For a hot-plug capable Downstream Port (as indicated by the Hot-Plug Capable bit of the Slot Capabilities Register) or a Downstream Port that supports Link speeds greater than 5.0 GT/s, this bit must be hardwired to 1b.</p> <p>For Upstream Ports and components that do not support this optional capability, this bit must be hardwired to 0b.</p> | RO |
| 21 | <p>Link Bandwidth Notification Capability - A value of 1b indicates support for the Link Bandwidth Notification status and interrupt mechanisms. This capability is required for all Root Ports and Switch Downstream Ports supporting Links wider than x1 and/or multiple Link speeds.</p> <p>This field is not applicable and is Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches.</p> <p>Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b.</p> | RO |
| 22 | <p>ASPM Optionality Compliance - This bit must be set to 1b in all Functions. Components implemented against certain earlier versions of this specification will have this bit set to 0b.</p> <p>Software is permitted to use the value of this bit to help determine whether to enable ASPM or whether to run ASPM compliance tests.</p> | HwInit |
| 31:24 | <p>Port Number - This field indicates the PCI Express Port number for the given PCI Express Link.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | HwInit |

IMPLEMENTATION NOTE: USE OF THE ASPM OPTIONALITY COMPLIANCE BIT §

Correct implementation and utilization of ASPM can significantly reduce Link power. However, ASPM feature implementations can be complex, and historically, some implementations have not been compliant to the specification. To address this, some of the ASPM optionality and ASPM entry requirements from earlier revisions of this document have been loosened. However, clear pass/fail compliance testing for ASPM features is also supported and expected.

The ASPM Optionality Compliance bit was created as a tool to establish clear expectations for hardware and software. This bit is Set to indicate hardware that conforms to the current specification, and this bit must be Set in components compliant to this specification.

System software as well as compliance software can assume that if this bit is Set, that the associated hardware conforms to the current specification. Hardware should be fully capable of supporting ASPM configuration management without needing component-specific treatment by system software.

For older hardware that does not have this bit Set, it is strongly recommended for system software to provide mechanisms to enable ASPM on components that work correctly with ASPM, and to disable ASPM on components that don't.

7.5.3.7 Link Control Register (Offset 10h) §

The Link Control Register controls PCI Express Link specific parameters. § Figure 7-28 details allocation of register fields in the Link Control Register ; § Table 7-24 provides the respective bit definitions.

For VF fields indicated as RsvdP , the associated PF's setting applies to the VF.

Base 6.4 vs Base 6.3

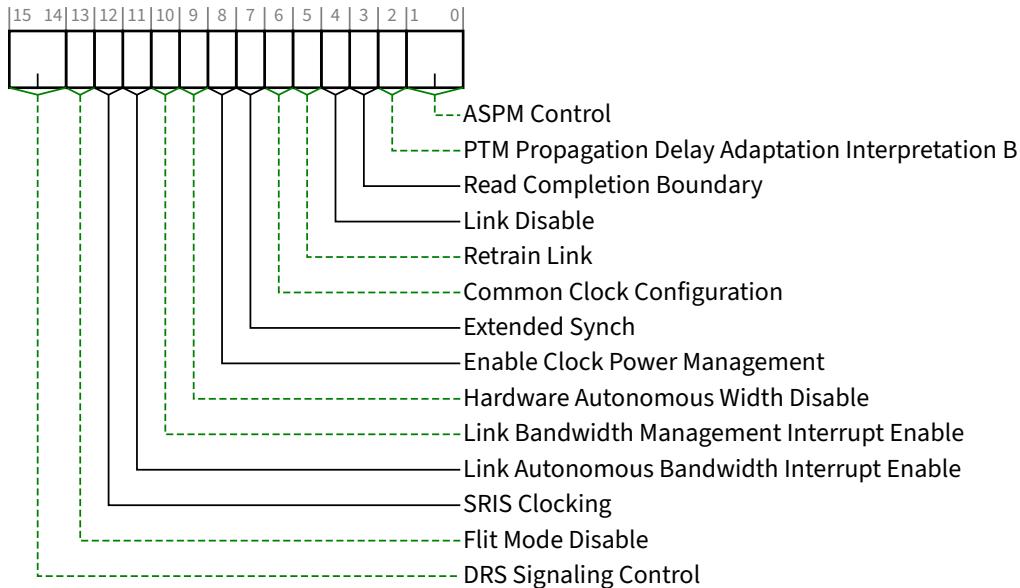


Figure 7-28 Link Control Register §

Table 7-24 Link Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 1:0 | <p>ASPM Control / Active State Power Management Control - This field controls the level of ASPM enabled on the given PCI Express Link. See § Section 5.4.1.4 for requirements on when and how to enable ASPM.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b Disabled 01b L0s Entry Enabled 10b L1 Entry Enabled 11b L0s and L1 Entry Enabled <p>Note: “L0s Entry Enabled” enables the Transmitter to enter L0s . If L0s is supported, the Receiver must be capable of entering L0s even when the Transmitter is disabled from entering L0s (00b or 10b).</p> <p>In Flit Mode, L0s is not supported, bit 0 of this field is ignored and has no effect (i.e., encodings 01b and 00b are equivalent as are encodings 11b and 10b).</p> <p>ASPM L1 must be enabled by software in the Upstream component on a Link prior to enabling ASPM L1 in the Downstream component on that Link. When disabling ASPM L1, software must disable ASPM L1 in the Downstream component on a Link prior to disabling ASPM L1 in the Upstream component on that Link. ASPM L1 must only be enabled on the Downstream component if both components on a Link support ASPM L1.</p> <p>For Multi-Function Devices (including ARI Devices), it is recommended that software program the same value for this field in all Functions. For non-ARI Multi-Function Devices , only capabilities enabled in all Functions are enabled for the component as a whole.</p> <p>For ARI Devices , ASPM Control is determined solely by the setting in Function 0, regardless of Function 0's D-state. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component.</p> | RW VF RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|---|
| | Default value of this field is 00b unless otherwise required by a particular form factor. | |
| 2 | <p>PTM Propagation Delay Adaptation Interpretation B – For a device that supports PTM, if PTM Propagation Delay Adaptation Capable in the PTM Capability Register is Set, then, for an Upstream Port, this bit when Set selects interpretation B of the Propagation Delay[31:0] field for received PTM ResponseD Messages , and for a Downstream Port, this bit when Set selects interpretation B of the Propagation Delay[31:0] field for PTM ResponseD Messages transmitted by the Port; otherwise this bit when Clear selects interpretation A for both cases. For a device that supports PTM, if its PTM Propagation Delay Adaptation Capable in the PTM Capability Register is Clear, Ports must hardwire this bit to 0b. For Multi-Function Devices associated with an Upstream Port of a device that supports PTM, this bit must be implemented in the same Function that contains the PTM Extended Capability structure and RsvdP in all other Functions. Default value is implementation specific, but is recommended to be 0b. For a device that does not support PTM, for all Ports in that device this bit must be RsvdP .</p> | RW / RsvdP |
| 3 | <p>Read Completion Boundary (RCB) - field is meaningful in Root Ports, Endpoints and Bridges. When meaningful, defined encodings are:</p> <p>0b 64 byte 1b 128 byte</p> <p>Root Ports: RCB contains the RCB value for the Root Port. Refer to § Section 2.3.1.1 for the definition of the parameter RCB .</p> <p>This bit is hardwired for a Root Port and returns its RCB support capabilities.</p> <p>Endpoints and Bridges: Read Completion Boundary (RCB) - Optionally Set by configuration software to indicate the RCB value of the Root Port Upstream from the Endpoint or Bridge. Refer to § Section 2.3.1.1 for the definition of the parameter RCB .</p> <p>Configuration software must only Set this bit if the Root Port Upstream from the Endpoint or Bridge reports an RCB value of 128 bytes (a value of 1b in the Read Completion Boundary bit).</p> <p>Default value of this bit is 0b.</p> <p>Functions that do not implement this feature must hardwire the bit to 0b.</p> <p>Switch Ports: Not applicable - must hardwire the bit to 0b</p> | Root Ports: RO Endpoints and Bridges: RW VF RsvdP Switch Ports: RO |
| 4 | <p>Link Disable - This bit disables the Link by directing the LTSSM to the Disabled state when Set; this bit is Reserved on Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches.</p> <p>See Implementation Note: Delays in Data Link Layer Link Active Reflecting Link Control Operations for related information.</p> <p>Writes to this bit are immediately reflected in the value read from the bit, regardless of actual Link state.</p> <p>After clearing this bit, software must honor timing requirements defined in § Section 6.6.1 with respect to the first Configuration Read following a Conventional Reset.</p> <p>Default value of this bit is 0b.</p> | RW |
| 5 | <p>Retrain Link - A write of 1b to this bit initiates Link retraining by directing the Physical Layer LTSSM to the Recovery state. If the LTSSM is already in Recovery or Configuration, re-entering Recovery is permitted but not required. If the Port is in DPC when a write of 1b to this bit occurs, the result is undefined. Reads of this bit always return 0b.</p> <p>It is permitted to write 1b to this bit while simultaneously writing modified values to other fields in this register. If the LTSSM is not already in Recovery or Configuration, the resulting Link training must</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------------|
| | <p>use the modified values. If the LTSSM is already in Recovery or Configuration, the modified values are not required to affect the Link training that's already in progress.</p> <p>This bit is not applicable and is Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches.</p> <p>This bit always returns 0b when read.</p> | |
| 6 | <p>Common Clock Configuration - When Set, this bit indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock.</p> <p>A value of 0b indicates that this component and the component at the opposite end of this Link are operating with asynchronous reference clock.</p> <p>For non-ARI Multi-Function Devices , software must program the same value for this bit in all Functions. If not all Functions are Set, then the component must as a whole assume that its reference clock is not common with the Upstream component.</p> <p>For ARI Devices , Common Clock Configuration is determined solely by the setting in Function 0. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component.</p> <p>Components utilize this <u>Common Clock Configuration</u> information to report the correct <u>L0s</u> and <u>L1 Exit</u> Latencies.</p> <p>After changing the value in this bit in both components on a Link, software must trigger the Link to retrain by writing a 1b to the <u>Retrain Link</u> bit of the Downstream Port.</p> <p>Default value of this bit is 0b.</p> | <u>RW</u> <u>VF RsvdP</u> |
| 7 | <p>Extended Synch - When Set, this bit forces the transmission of additional Ordered Sets when exiting the <u>L0s</u> state (see § Section 4.2.5.6) and when in the Recovery state (see § Section 4.2.7.4.1). This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and Symbol lock before the Link enters the L0 state and resumes communication.</p> <p>For Multi-Function Devices if any Function has this bit Set, then the component must transmit the additional Ordered Sets when exiting <u>L0s</u> or when in Recovery.</p> <p>Default value for this bit is 0b.</p> | <u>RW</u> <u>VF RsvdP</u> |
| 8 | <p>Enable Clock Power Management - Applicable only for Upstream Ports and with form factors that support a “Clock Request” (CLKREQ#) mechanism, this bit operates as follows:</p> <ul style="list-style-type: none"> 0b Clock power management is disabled and device must hold CLKREQ# signal low. 1b When this bit is Set, the device is permitted to use CLKREQ# signal to power manage Link clock according to protocol defined in appropriate form factor specification. <p>For a non-ARI Multi-Function Device , power-management-configuration software must only Set this bit if all Functions of the Multi-Function Device indicate a 1b in the <u>Clock Power Management</u> bit of the <u>Link Capabilities Register</u> . The component is permitted to use the CLKREQ# signal to power manage Link clock only if this bit is Set for all Functions.</p> <p>For ARI Devices , Clock Power Management is enabled solely by the setting in Function 0. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component.</p> <p>The CLKREQ# signal may also be controlled via the <u>L1 PM Substates</u> mechanism. Such control is not affected by the setting of this bit.</p> <p>Downstream Ports and components that do not support Clock Power Management (as indicated by a 0b value in the <u>Clock Power Management</u> bit of the <u>Link Capabilities Register</u>) must hardwire this bit to 0b.</p> <p>Default value of this bit is 0b, unless specified otherwise by the form factor specification.</p> | <u>RW</u> <u>VF RsvdP</u> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | |
|---------------|--|---|----------------------------|---------------|--------------|---|---|------|---|---|------|---|---|----|
| 9 | <p>Hardware Autonomous Width Disable - When Set, this bit disables hardware from changing the Link width for reasons other than attempting to correct unreliable Link operation by reducing Link width.</p> <p>For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RW , and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP .</p> <p>Components that do not implement the ability autonomously to change Link width are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW / RsvdP (see description) VF RsvdP | | | | | | | | | | | | |
| 10 | <p>Link Bandwidth Management Interrupt Enable - When Set, this bit enables the generation of an interrupt to indicate that the Link Bandwidth Management Status bit has been Set.</p> <p>This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches.</p> <p>Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | | | | | |
| 11 | <p>Link Autonomous Bandwidth Interrupt Enable - When Set, this bit enables the generation of an interrupt to indicate that the Link Autonomous Bandwidth Status bit has been Set.</p> <p>This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches.</p> <p>Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | | | | | |
| 12 | <p>SRIS Clocking - This bit, in conjunction with Common Clock Configuration , indicates the clocking mode used on the Link.</p> <p>This bit is meaningful in Downstream Ports that support Flit Mode . In all other Functions, this bit is RsvdP .</p> <p>If Common Clock Configuration is Set, this bit has no effect and the SRIS Clocking bit in the TS1s must be 0b (Symbol 4, bit 7).</p> <p>If Common Clock Configuration is Clear, this bit is sent in the SRIS Clocking bit of TS1s (Symbol 4, bit 7).</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Clocking Mode</th> <th>Common Clock Configuration</th> <th>SRIS Clocking</th> </tr> </thead> <tbody> <tr> <td>Common Clock</td> <td>1</td> <td>x</td> </tr> <tr> <td>SRNS</td> <td>0</td> <td>0</td> </tr> <tr> <td>SRIS</td> <td>0</td> <td>1</td> </tr> </tbody> </table> <p>Default is 0b.</p> | Clocking Mode | Common Clock Configuration | SRIS Clocking | Common Clock | 1 | x | SRNS | 0 | 0 | SRIS | 0 | 1 | RW |
| Clocking Mode | Common Clock Configuration | SRIS Clocking | | | | | | | | | | | | |
| Common Clock | 1 | x | | | | | | | | | | | | |
| SRNS | 0 | 0 | | | | | | | | | | | | |
| SRIS | 0 | 1 | | | | | | | | | | | | |
| 13 | <p>Flit Mode Disable - when Set, the Port is not permitted to set the Flit Mode Supported bit in training sets it sends. This bit has no effect on the Flit Mode Supported bit in the PCI Express Capabilities Register and thus has no effect on behavior required by MUST@FLIT.</p> <p>Since Flit Mode is required at 64.0 GT/s or higher, disabling Flit Mode also has the effect of disabling data rates of 64.0 GT/s or higher.</p> <p>This bit is mandatory in Downstream Ports where Flit Mode Supported is Set.</p> <p>For Functions associated with an Upstream Port, this bit is optionally implemented in Function 0 and is not implemented in all other Functions. When not implemented, this bit must be hardwired to Zero .</p> <p>Changing this bit while the Link is Up has no effect. The updated value will take effect on the next transition to Link Up.</p> | RW | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>This bit can be used as a workaround for faulty Flit Mode implementations. As such, it might be set by System Firmware, Device Firmware. As such, System Software should not clear this bit.</p> <p>This bit is not implemented in RCiEPs.</p> <p>In Downstream Ports, the default is Zero . In Upstream Ports, the default is implementation specific (e.g., it might be Set by device firmware).</p> | |
| 15:14 | <p>DRS Signaling Control - Indicates the mechanism used to report reception of a DRS message. Must be implemented for Downstream Ports with the <u>DRS Supported</u> bit Set in the <u>Link Capabilities 2 Register</u>. Encodings are:</p> <ul style="list-style-type: none"> 00b DRS not Reported: If DRS Supported is Set, receiving a DRS Message will set <u>DRS Message Received</u> in the <u>Link Status 2 Register</u> but will otherwise have no effect 01b DRS Interrupt Enabled: If the <u>DRS Message Received</u> bit in the <u>Link Status 2 Register</u> transitions from 0 to 1, and interrupts are enabled, an interrupt must be generated. If either MSI or MSI-X is enabled, an MSI or MSI-X interrupt is generated using the vector in <u>Interrupt Message Number</u> (§ Section 7.5.3.2) 10b DRS to FRS Signaling Enabled: If the <u>DRS Message Received</u> bit in the <u>Link Status 2 Register</u> transitions from 0 to 1, the Port must send an FRS Message Upstream with the FRS Reason field set to DRS Message Received . <p>Behavior is undefined if this field is set to 10b and the <u>FRS Supported</u> bit in the <u>Device Capabilities 2 Register</u> is Clear.</p> <p>Behavior is undefined if this field is set to 11b.</p> <p>Downstream Ports with the <u>DRS Supported</u> bit Clear in the <u>Link Capabilities 2 Register</u> must hardwire this field to 00b.</p> <p>This field is Reserved for Upstream Ports.</p> <p>Default value of this field is 00b.</p> | RW / RsvdP |

IMPLEMENTATION NOTE: SOFTWARE COMPATIBILITY WITH ARI DEVICES §

With the ASPM Control field, Common Clock Configuration bit, and Enable Clock Power Management bit in the Link Control Register, there are potential software compatibility issues with ARI Devices since these controls operate strictly off the settings in Function 0 instead of the settings in all Functions.

With compliant software, there should be no issues with the Common Clock Configuration bit, since software is required to set this bit the same in all Functions.

With the Enable Clock Power Management bit, there should be no compatibility issues with software that sets this bit the same in all Functions. However, if software does not set this bit the same in all Functions, and relies on each Function having the ability to prevent Clock Power Management from being enabled, such software may have compatibility issues with ARI Devices.

With the ASPM Control field, there should be no compatibility issues with software that sets this bit the same in all Functions. However, if software does not set this bit the same in all Functions, and relies on each Function in D0 state having the ability to prevent ASPM from being enabled, such software may have compatibility issues with ARI Devices.

IMPLEMENTATION NOTE: AVOIDING RACE CONDITIONS WHEN USING THE RETRAIN LINK BIT §

When software changes Link control parameters and writes a 1b to the Retrain Link bit in order to initiate Link training using the new parameter settings, special care is required in order to avoid certain race conditions. At any instant the LTSSM may transition to the Recovery or Configuration state due to normal Link activity, without software awareness. If the LTSSM is already in Recovery or Configuration when software writes updated parameters to the Link Control Register, as well as a 1b, to the Retrain Link bit, the LTSSM might not use the updated parameter settings with the current Link training, and the current Link training might not achieve the results that software intended.

To avoid this potential race condition, it is strongly recommended that software use the following algorithm or something similar:

1. Software sets the relevant Link control parameters to the desired settings without writing a 1b to the Retrain Link bit.
2. Software polls the Link Training bit in the Link Status Register until the value returned is 0b.
3. Software writes a 1b to the Retrain Link bit without changing any other fields in the Link Control Register.

The above algorithm guarantees that Link training will be based on the Link control parameter settings that software intends.

IMPLEMENTATION NOTE: USE OF THE SLOT CLOCK CONFIGURATION AND COMMON CLOCK CONFIGURATION BITS §

In order to determine the common clocking configuration of components on opposite ends of a Link that crosses a connector, two pieces of information are required. The following description defines these requirements.

The first necessary piece of information is whether the Downstream Port that connects to the slot uses a clock that has a common source and therefore constant phase relationship to the clock signal provided on the slot. This information is provided by the system side component through a hardware initialized bit (Slot Clock Configuration) in its Link Status Register. Note that some electromechanical form factor specifications may require the Port that connects to the slot use a clock that has a common source to the clock signal provided on the slot.

The second necessary piece of information is whether the component on the adapter uses the clock supplied on the slot or one generated locally on the adapter. The adapter design and layout will determine whether the component is connected to the clock source provided by the slot. A component going onto this adapter should have some hardware initialized method for the adapter design/designer to indicate the configuration used for this particular adapter design. This information is reported by Slot Clock Configuration in the Link Status Register of each Function in the Upstream Port. Note that some electromechanical form factor specifications may require the Port on the adapter to use the clock signal provided on the connector.

System firmware or software will read the Slot Clock Configuration from the components on both ends of a physical Link. If the Slot Clock Configuration bit is Set for both components, this firmware/software will Set the Common Clock Configuration bit on both components connected to the Link. Each component uses this bit to determine the length of time required to re-synch its Receiver to the opposing component's Transmitter when exiting L0s.

The time required to re-synch is reported as a time value in the L0s Exit Latency in the Link Capabilities Register (offset 0Ch) and is sent to the opposing Transmitter as part of the initialization process as N_FTS. Components would be expected to require much longer synch times without common clocking and would therefore report a longer L0s Exit Latency in bits 12-14 of the Link Capabilities Register and would send a larger number for N_FTS during training. This forces a requirement that whatever software changes this bit should force a Link retrain in order to get the correct N_FTS set for the Receivers at both ends of the Link.

7.5.3.8 Link Status Register (Offset 12h) §

The Link Status Register provides information about PCI Express Link specific parameters. § Figure 7-29 details allocation of register fields in the Link Status Register; § Table 7-26 provides the respective bit definitions.

For a VF, all fields are RsvdZ and the associated PF's setting for each field applies to the VF.

Base 6.4 vs Base 6.3

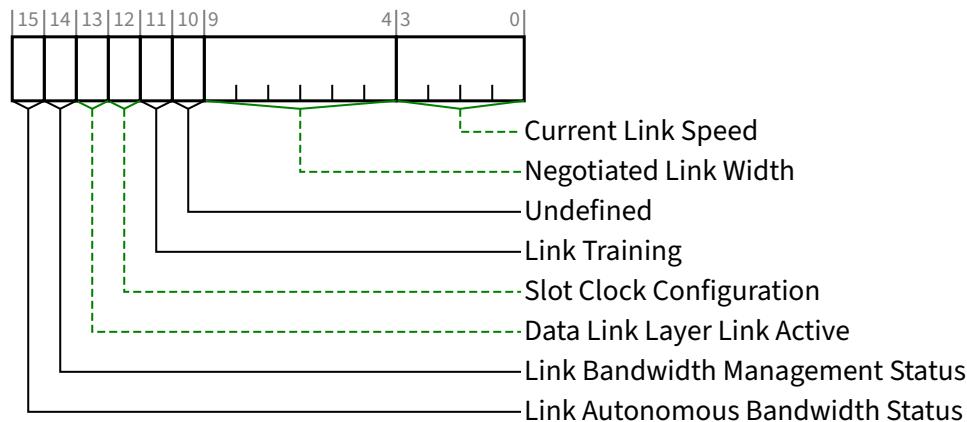


Figure 7-29 Link Status Register

Table 7-26 Link Status Register

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------|
| 3:0 | <p>Current Link Speed - This field indicates the negotiated Link speed of the given PCI Express Link. The encoded value specifies a Bit Location in the Supported Link Speeds Vector (in the Link Capabilities 2 Register) that corresponds to the current Link speed.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 <p>All other encodings are Reserved.</p> <p>The value in this field is undefined when the Link is not up.</p> | <p>RO VF RsvdZ</p> |
| 9:4 | <p>Negotiated Link Width - This field indicates the negotiated width of the given PCI Express Link. This includes the Link Width determined during initial link training as well changes that occur after initial link training (e.g., L0p).</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 01 0000b x16 <p>All other encodings are Reserved. The value in this field is undefined when the Link is not up.</p> | <p>RO VF RsvdZ</p> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|----------------------------------|
| 10 | Undefined - The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit. | <u>RO</u> <u>VF RsvdZ</u> |
| 11 | Link Training - This read-only bit indicates that the Physical Layer LTSSM is in the Configuration or Recovery state, or that 1b was written to the Retrain Link bit but Link training has not yet begun. Hardware clears this bit when the LTSSM exits the Configuration/Recovery state. This bit is not applicable and Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches, and must be hardwired to 0b. | <u>RO</u> <u>VF RsvdZ</u> |
| 12 | Slot Clock Configuration - This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference clock on the connector, this bit must be clear. For a Multi-Function Device , each Function must report the same value for this bit. | <u>HwInit</u> <u>VF RsvdZ</u> |
| 13 | Data Link Layer Link Active - This bit indicates the status of the Data Link Control and Management State Machine. It returns a 1b to indicate the <u>DL_Active</u> state, 0b otherwise. See Implementation Note: Delays in Data Link Layer Link Active Reflecting Link Control Operations for related information. This bit must be implemented if the <u>Data Link Layer Link Active Reporting Capable</u> bit is 1b. Otherwise, this bit must be hardwired to 0b. | <u>RO</u> <u>VF RsvdZ</u> |
| 14 | Link Bandwidth Management Status - This bit is Set by hardware to indicate that either of the following has occurred without the Port transitioning through <u>DL_Down</u> status: <ul style="list-style-type: none"> A Link retraining has completed following a write of 1b to the <u>Retrain Link</u> bit. Note: This bit is Set following any write of 1b to the <u>Retrain Link</u> bit, including when the Link is in the process of retraining for some other reason. Hardware has changed Link speed or width to attempt to correct unreliable Link operation, either through an LTSSM timeout or a higher level process. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was not indicated as an autonomous change. This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the <u>Link Bandwidth Notification Capability</u> must hardwire this bit to 0b. The default value of this bit is 0b. | <u>RW1C</u> <u>VF RsvdZ</u> |
| 15 | Link Autonomous Bandwidth Status - This bit is Set by hardware to indicate that hardware has autonomously changed Link speed or width, without the Port transitioning through <u>DL_Down</u> status, for reasons other than to attempt to correct unreliable Link operation. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was indicated as an autonomous change. This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the <u>Link Bandwidth Notification Capability</u> must hardwire this bit to 0b. The default value of this bit is 0b. | <u>RW1C</u> <u>VF RsvdZ</u> |

IMPLEMENTATION NOTE: DELAYS IN DLL LINK ACTIVE REFLECTING LINK CONTROL OPERATIONS §

When software changes Link control parameters such as Setting the Secondary Bus Reset bit in the Bridge Control Register or the Link Disable bit in the Link Control Register, the Downstream Port will eventually transition to the DL_Down state, but there may be a significant delay with this occurring and being reflected by the Data Link Layer Link Active bit in the Link Status Register being Cleared. Often this occurs within a few ms, but in certain cases it may take tens of ms or longer. When software is waiting for Data Link Layer Link Active to become Clear, in some environments it may be best for software to set up a Data Link Layer State Changed interrupt instead of polling Data Link Layer Link Active continuously until it Clears

7.5.3.9 Slot Capabilities Register (Offset 14h) §

The Slot Capabilities Register identifies PCI Express slot specific capabilities. § Figure 7-30 details allocation of register fields in the Slot Capabilities Register; § Table 7-27 provides the respective bit definitions.

If this register is implemented but the Slot Implemented bit is Clear, the field behavior of this entire register is undefined.

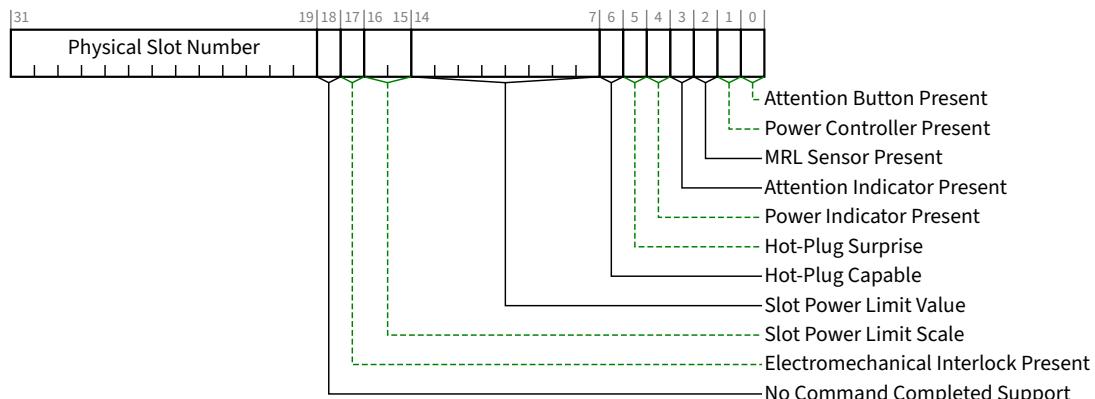


Figure 7-30 Slot Capabilities Register §

Table 7-27 Slot Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | Attention Button Present - When Set, this bit indicates that an Attention Button for this slot is electrically controlled by the chassis. | HwInit |
| 1 | Power Controller Present - When Set, this bit indicates that a software programmable Power Controller is implemented for this slot/adapter (depending on form factor). | HwInit |
| 2 | MRL Sensor Present - When Set, this bit indicates that an MRL Sensor is implemented on the chassis for this slot. | HwInit |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|---|---|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|---------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--|---------------|
| 3 | Attention Indicator Present - When Set, this bit indicates that an Attention Indicator is electrically controlled by the chassis. | <u>HwInit</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | Power Indicator Present - When Set, this bit indicates that a Power Indicator is electrically controlled by the chassis for this slot. | <u>HwInit</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Hot-Plug Surprise - When Set, this bit indicates that an adapter present in this slot might be removed from the system without any prior notification. This is a form factor specific capability. This bit is an indication to the operating system to allow for such removal without impacting continued software operation. If the SFI HPS Suppress bit in the SFI Control Register is Clear, a read of the Hot-Plug Surprise bit returns the <u>HwInit</u> value. If the SFI HPS Suppress bit is Set, a read returns 0b. See [4] See § Section 7.9.22.3 . | <u>HwInit / RO</u> (see description) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | Hot-Plug Capable - When Set, this bit indicates that this slot is capable of supporting hot-plug operations. | <u>HwInit</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14:7 | <p>Slot Power Limit Value - In combination with the <u>Slot Power Limit Scale</u> value, specifies the upper limit on power supplied by the slot (<u>see § Section 6.9</u>) or by other means to the adapter.</p> <p>Power limit (in Watts) is calculated by multiplying the value in this field by the value in the <u>Slot Power Limit Scale</u> field except when the <u>Slot Power Limit Scale</u> field equals 00b (1.0x) and <u>Slot Power Limit Value</u> exceeds EFh, the following alternative encodings are used:</p> <table> <tbody> <tr><td>F0h</td><td>> 239 W and ≤ 250 W Slot Power Limit</td></tr> <tr><td>F1h</td><td>> 250 W and ≤ 275 W Slot Power Limit</td></tr> <tr><td>F2h</td><td>> 275 W and ≤ 300 W Slot Power Limit</td></tr> <tr><td>F3h</td><td>> 300 W and ≤ 325 W Slot Power Limit</td></tr> <tr><td>F4h</td><td>> 325 W and ≤ 350 W Slot Power Limit</td></tr> <tr><td>F5h</td><td>> 350 W and ≤ 375 W Slot Power Limit</td></tr> <tr><td>F6h</td><td>> 375 W and ≤ 400 W Slot Power Limit</td></tr> <tr><td>F7h</td><td>> 400 W and ≤ 425 W Slot Power Limit</td></tr> <tr><td>F8h</td><td>> 425 W and ≤ 450 W Slot Power Limit</td></tr> <tr><td>F9h</td><td>> 450 W and ≤ 475 W Slot Power Limit</td></tr> <tr><td>FAh</td><td>> 475 W and ≤ 500 W Slot Power Limit</td></tr> <tr><td>FBh</td><td>> 500 W and ≤ 525 W Slot Power Limit</td></tr> <tr><td>FCh</td><td>> 525 W and ≤ 550 W Slot Power Limit</td></tr> <tr><td>FDh</td><td>> 550 W and ≤ 575 W Slot Power Limit</td></tr> <tr><td>FEh</td><td>> 575 W and ≤ 600 W Slot Power Limit</td></tr> <tr><td>FFh</td><td>Reserved for Slot Power Limit Values above 600 W</td></tr> </tbody> </table> <p>This register must be implemented if the <u>Slot Implemented</u> bit is Set.</p> <p>Writes to this register also cause the Port to send the <u>Set_Slot_Power_Limit Message</u>.</p> <p>The default value prior to hardware/firmware initialization is 0000 0000b.</p> | F0h | > 239 W and ≤ 250 W Slot Power Limit | F1h | > 250 W and ≤ 275 W Slot Power Limit | F2h | > 275 W and ≤ 300 W Slot Power Limit | F3h | > 300 W and ≤ 325 W Slot Power Limit | F4h | > 325 W and ≤ 350 W Slot Power Limit | F5h | > 350 W and ≤ 375 W Slot Power Limit | F6h | > 375 W and ≤ 400 W Slot Power Limit | F7h | > 400 W and ≤ 425 W Slot Power Limit | F8h | > 425 W and ≤ 450 W Slot Power Limit | F9h | > 450 W and ≤ 475 W Slot Power Limit | FAh | > 475 W and ≤ 500 W Slot Power Limit | FBh | > 500 W and ≤ 525 W Slot Power Limit | FCh | > 525 W and ≤ 550 W Slot Power Limit | FDh | > 550 W and ≤ 575 W Slot Power Limit | FEh | > 575 W and ≤ 600 W Slot Power Limit | FFh | Reserved for Slot Power Limit Values above 600 W | <u>HwInit</u> |
| F0h | > 239 W and ≤ 250 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F1h | > 250 W and ≤ 275 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F2h | > 275 W and ≤ 300 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F3h | > 300 W and ≤ 325 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F4h | > 325 W and ≤ 350 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F5h | > 350 W and ≤ 375 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F6h | > 375 W and ≤ 400 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F7h | > 400 W and ≤ 425 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F8h | > 425 W and ≤ 450 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F9h | > 450 W and ≤ 475 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FAh | > 475 W and ≤ 500 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FBh | > 500 W and ≤ 525 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FCh | > 525 W and ≤ 550 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDh | > 550 W and ≤ 575 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FEh | > 575 W and ≤ 600 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FFh | Reserved for Slot Power Limit Values above 600 W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16:15 | <p>Slot Power Limit Scale - Specifies the scale used for the <u>Slot Power Limit Value</u> (<u>see § Section 6.9</u>).</p> <p>Range of Values:</p> <table> <tbody> <tr><td>00b</td><td>1.0x</td></tr> <tr><td>01b</td><td>0.1x</td></tr> <tr><td>10b</td><td>0.01x</td></tr> <tr><td>11b</td><td>0.001x</td></tr> </tbody> </table> | 00b | 1.0x | 01b | 0.1x | 10b | 0.01x | 11b | 0.001x | <u>HwInit</u> | | | | | | | | | | | | | | | | | | | | | | | | |
| 00b | 1.0x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01b | 0.1x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10b | 0.01x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11b | 0.001x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| | <p>This register must be implemented if the <u>Slot Implemented</u> bit is Set.</p> <p>Writes to this register also cause the Port to send the <u>Set_Slot_Power_Limit Message</u>.</p> <p>The default value prior to hardware/firmware initialization is 00b.</p> | |
| 17 | Electromechanical Interlock Present - When Set, this bit indicates that an Electromechanical Interlock is implemented on the chassis for this slot. | <u>HwInit</u> |
| 18 | No Command Completed Support - When Set, this bit indicates that this slot does not generate software notification when an issued command is completed by the Hot-Plug Controller. This bit is only permitted to be Set if the hot-plug capable Port is able to accept writes to all fields of the <u>Slot Control Register</u> without delay between successive writes. | <u>HwInit</u> |
| 31:19 | Physical Slot Number - This field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is unique within the chassis, regardless of the form factor associated with the slot. This field must be initialized to Zero for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port. | <u>HwInit</u> |

7.5.3.10 Slot Control Register (Offset 18h) §

The Slot Control Register controls PCI Express Slot specific parameters. § Figure 7-31 details allocation of register fields in the Slot Control Register; § Table 7-28 provides the respective bit definitions.

Attention Indicator Control , Power Indicator Control , and Power Controller Control fields of the Slot Control Register do not have a defined default value. If these fields are implemented, it is the responsibility of either system firmware or operating system software to (re)initialize these fields after a reset of the Link.

In hot-plug capable Downstream Ports, a write to the Slot Control Register must cause a hot-plug command to be generated (see § Section 6.7.3.2 for details on hot-plug commands). A write to the Slot Control Register in a Downstream Port that is not hot-plug capable must not cause a hot-plug command to be executed.

If this register is implemented but the Slot Implemented bit is Clear, the field behavior of this entire register with the exception of the Data Link Layer State Changed Enable bit is undefined.

Base 6.4 vs Base 6.3

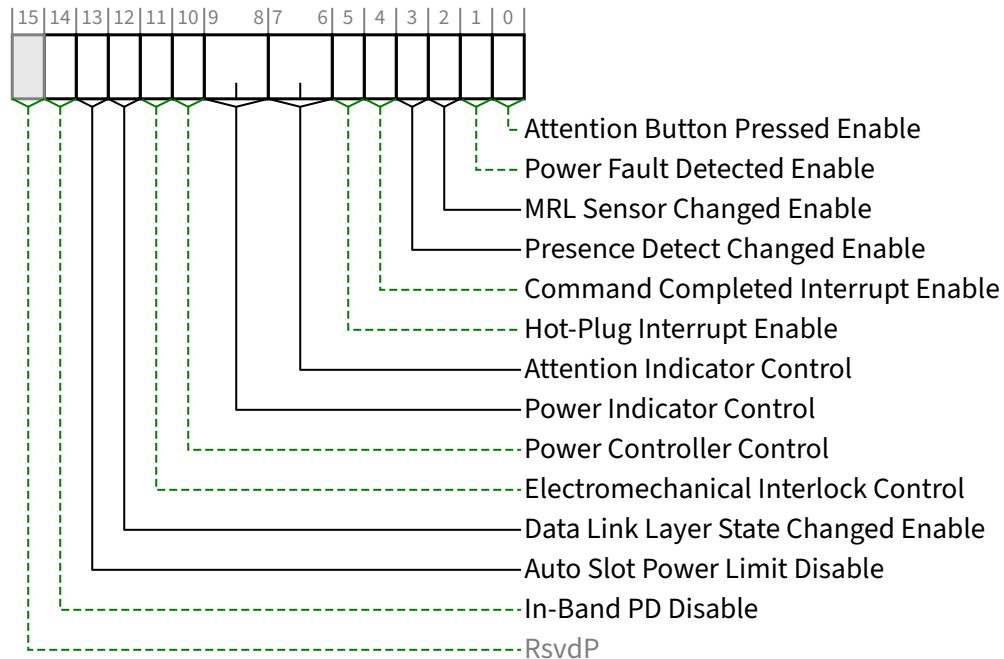


Figure 7-31 Slot Control Register §

Table 7-28 Slot Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Attention Button Pressed Enable - When Set to 1b, this bit enables software notification on an attention button pressed event (see § Section 6.7.3).</p> <p>If the Attention Button Present bit in the Slot Capabilities Register is 0b, this bit is permitted to be read-only with a value of 0b.</p> <p>Default value of this bit is 0b.</p> | RW |
| 1 | <p>Power Fault Detected Enable - When Set, this bit enables software notification on a power fault event (see § Section 6.7.3).</p> <p>If a Power Controller that supports power fault detection is not implemented, this bit is permitted to be read-only with a value of 0b.</p> <p>Default value of this bit is 0b.</p> | RW |
| 2 | <p>MRL Sensor Changed Enable - When Set, this bit enables software notification on a MRL sensor changed event (see § Section 6.7.3).</p> <p>If the MRL Sensor Present bit in the Slot Capabilities Register is Clear, this bit is permitted to be read-only with a value of 0b.</p> <p>Default value of this bit is 0b.</p> | RW |
| 3 | <p>Presence Detect Changed Enable - When Set, this bit enables software notification on a presence detect changed event (see § Section 6.7.3).</p> <p>If the Hot-Plug Capable bit in the Slot Capabilities Register is 0b, this bit is permitted to be read-only with a value of 0b.</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|--|------------|----------|------------|----|------------|-------|------------|-----|----|
| | Default value of this bit is 0b. | | | | | | | | | |
| 4 | <p>Command Completed Interrupt Enable - If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities Register is 0b), when Set, this bit enables software notification when a hot-plug command is completed by the Hot-Plug Controller.</p> <p>If Command Completed notification is not supported, this bit must be hardwired to 0b.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | |
| 5 | <p>Hot-Plug Interrupt Enable - When Set, this bit enables generation of an interrupt on enabled hot-plug events.</p> <p>If the Hot-Plug Capable bit in the Slot Capabilities Register is Clear, this bit is permitted to be read-only with a value of 0b.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | |
| 7:6 | <p>Attention Indicator Control - If an Attention Indicator is implemented, writes to this field set the Attention Indicator to the written state.</p> <p>Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting, if required to, for the previous command to complete in which case the read value is undefined.</p> <p>Defined encodings are:</p> <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> <p>Note: The default value of this field must be one of the non-Reserved values. If the Attention Indicator Present bit in the Slot Capabilities Register is 0b, this bit is permitted to be read-only with a value of 00b.</p> | 00b | Reserved | 01b | On | 10b | Blink | 11b | Off | RW |
| 00b | Reserved | | | | | | | | | |
| 01b | On | | | | | | | | | |
| 10b | Blink | | | | | | | | | |
| 11b | Off | | | | | | | | | |
| 9:8 | <p>Power Indicator Control - If a Power Indicator is implemented, writes to this field set the Power Indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting, if required to, for the previous command to complete in which case the read value is undefined.</p> <p>Defined encodings are:</p> <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> <p>Note: The default value of this field must be one of the non-Reserved values. If the Power Indicator Present bit in the Slot Capabilities Register is 0b, this bit is permitted to be read-only with a value of 00b.</p> | 00b | Reserved | 01b | On | 10b | Blink | 11b | Off | RW |
| 00b | Reserved | | | | | | | | | |
| 01b | On | | | | | | | | | |
| 10b | Blink | | | | | | | | | |
| 11b | Off | | | | | | | | | |
| 10 | <p>Power Controller Control - If a Power Controller is implemented, this bit when written sets the power state of the slot per the defined encodings. Reads of this bit must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write, if required to, without waiting for the previous command to complete in which case the read value is undefined.</p> <p>Note that in some cases the power controller may autonomously remove slot power or not respond to a power-up request based on a detected fault condition, independent of the Power Controller Control setting.</p> <p>The defined encodings are:</p> <table> <tr> <td>0b</td> <td>Power On</td> </tr> </table> | 0b | Power On | RW | | | | | | |
| 0b | Power On | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>1b Power Off If the Power Controller Present bit in the Slot Capabilities Register is Clear, then writes to this bit have no effect and the read value of this bit is undefined.</p> | |
| 11 | <p>Electromechanical Interlock Control - If an Electromechanical Interlock is implemented, a write of 1b to this bit causes the state of the interlock to toggle. A write of 0b to this bit has no effect. A read of this bit always returns a 0b.</p> | RW |
| 12 | <p>Data Link Layer State Changed Enable - If the Data Link Layer Link Active Reporting Capable is 1b, this bit enables software notification when Data Link Layer Link Active bit is changed. If the Data Link Layer Link Active Reporting Capable bit is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.</p> | RW |
| 13 | <p>Auto Slot Power Limit Disable - When Set, this disables the automatic sending of a Set_Slot_Power_Limit Message when a Link transitions from a non- DL_Up status to a DL_Up status. Downstream Ports that don't support DPC are permitted to hardwire this bit to 0. Default value of this bit is implementation specific.</p> | RW |
| 14 | <p>In-Band PD Disable - When Set, this bit disables the in-band presence detect mechanism from affecting the Presence Detect State bit, allowing that bit to report out-of-band presence detect exclusively. Otherwise, the Presence Detect State bit reflects the logical OR of the in-band and out-of-band presence detect mechanisms. In addition, the In-Band PD Disable bit governs the Component Presence state for the Downstream Component Presence field in the Link Status 2 Register . See § Section 7.5.3.20. This bit must be implemented if the In-Band PD Disable Supported bit is 1b. Otherwise, this bit must be hardwired to 0b. Default value of this bit is 0b.</p> | RW |

7.5.3.11 Slot Status Register (Offset 1Ah) §

The Slot Status Register provides information about PCI Express Slot specific parameters. § Figure 7-32 details allocation of register fields in the Slot Status Register ; § Table 7-29 provides the respective bit definitions. Register fields for status bits not implemented by the device have the RsvdZ attribute.

If this register is implemented but the Slot Implemented bit is Clear, the field behavior of this entire register with the exception of the Data Link Layer State Changed bit is undefined.

Base 6.4 vs Base 6.3

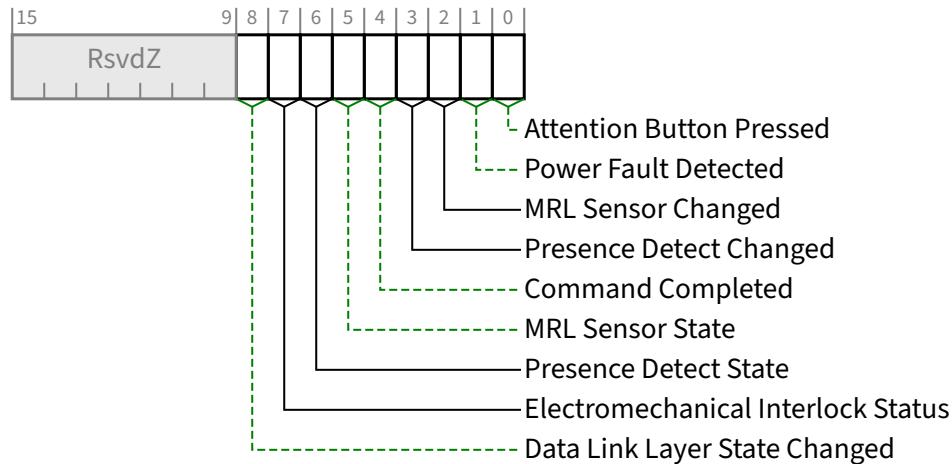


Figure 7-32 Slot Status Register §

Table 7-29 Slot Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Attention Button Pressed - If an Attention Button is implemented, this bit is Set when the attention button is pressed. If an Attention Button is not supported, this bit must not be Set. | RW1C |
| 1 | Power Fault Detected - If a Power Controller that supports power fault detection is implemented, this bit is Set when the Power Controller detects a power fault at this slot. Note that, depending on hardware capability, it is possible that a power fault can be detected at any time, independent of the Power Controller Control setting or the occupancy of the slot. If power fault detection is not supported, this bit must not be Set. | RW1C |
| 2 | MRL Sensor Changed - If an MRL sensor is implemented, this bit is Set when a <u>MRL Sensor State</u> change is detected. If an MRL sensor is not implemented, this bit must not be Set. | RW1C |
| 3 | Presence Detect Changed - This bit is Set when the value reported in the <u>Presence Detect State</u> bit is changed. | RW1C |
| 4 | Command Completed - If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities Register is 0b), this bit is Set when a hot-plug command has completed and the Hot-Plug Controller is ready to accept a subsequent command. The <u>Command Completed</u> status bit is Set as an indication to host software that the Hot-Plug Controller has processed the previous command and is ready to receive the next command; it provides no guarantee that the action corresponding to the command is complete. If Command Completed notification is not supported, this bit must be hardwired to 0b. | RW1C |
| 5 | MRL Sensor State - This bit reports the status of the MRL sensor if implemented. Defined encodings are: 0b MRL Closed 1b MRL Open | RO |
| 6 | Presence Detect State - This bit indicates the presence of an adapter in the slot. When the In-Band PD Disable bit is Clear, this is reflected by the logical “OR” of the Physical Layer in-band presence detect | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>mechanism and, if present, any out-of-band presence detect mechanism defined for the slot's corresponding form factor. Note that the in-band presence detect mechanism requires that power be applied to an adapter for its presence to be detected. Consequently, form factors that require a power controller for hot-plug must implement an out-of-band presence detect mechanism. When the <u>In-Band PD Disable</u> bit is Set, the in-band presence detect mechanism has no effect on this bit.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0b Adapter not Present 1b Adapter Present <p>This bit must be implemented on all Downstream Ports that implement slots. For Downstream Ports not connected to slots (where the <u>Slot Implemented</u> bit of the PCI Express Capabilities Register is 0b), this bit must be hardwired to 1b.</p> | |
| 7 | <p><i>Electromechanical Interlock Status</i> - If an Electromechanical Interlock is implemented, this bit indicates the status of the Electromechanical Interlock.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0b Electromechanical Interlock Disengaged 1b Electromechanical Interlock Engaged | RO |
| 8 | <p><i>Data Link Layer State Changed</i> - This bit is Set when the value reported in the <u>Data Link Layer Link Active</u> bit of the <u>Link Status Register</u> is changed.</p> <p>In response to a <u>Data Link Layer State Changed</u> event, software must read the <u>Data Link Layer Link Active</u> bit of the <u>Link Status Register</u> to determine if the Link is active before initiating configuration cycles to the hot plugged device.</p> | RW1C |

IMPLEMENTATION NOTE: NO SLOT POWER CONTROLLER §

For slots that do not implement a power controller, software must ensure that system power planes are enabled to provide power to slots prior to reading Presence Detect State.

7.5.3.12 Root Control Register (Offset 1Ch) §

The Root Control Register controls PCI Express Root Complex specific parameters. § Figure 7-33 details allocation of register fields in the Root Control Register; § Table 7-30 provides the respective bit definitions.

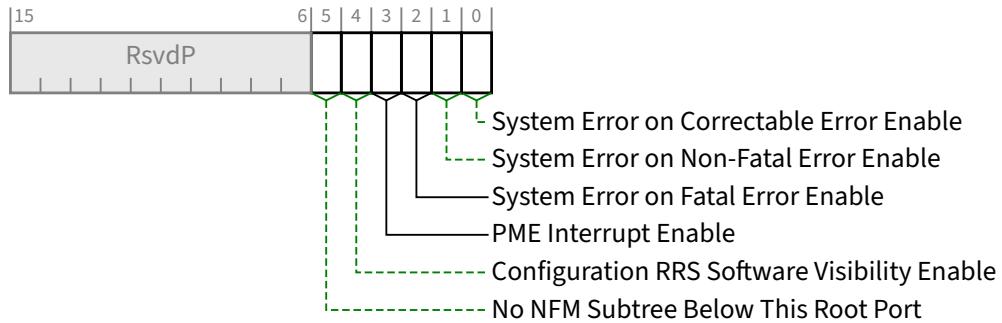


Figure 7-33 Root Control Register §

Table 7-30 Root Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>System Error on Correctable Error Enable - If Set, this bit indicates that a System Error should be generated if a correctable error (ERR_COR) is reported by any of the devices in the Hierarchy Domain associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific.</p> <p>Root Complex Event Collectors provide support for the above-described functionality for RCiEPs.</p> <p>Default value of this bit is 0b.</p> | RW |
| 1 | <p>System Error on Non-Fatal Error Enable - If Set, this bit indicates that a System Error should be generated if a Non-fatal error (ERR_NONFATAL) is reported by any of the devices in the Hierarchy Domain associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific.</p> <p>Root Complex Event Collectors provide support for the above-described functionality for RCiEPs.</p> <p>Default value of this bit is 0b.</p> | RW |
| 2 | <p>System Error on Fatal Error Enable - If Set, this bit indicates that a System Error should be generated if a Fatal error (ERR_FATAL) is reported by any of the devices in the Hierarchy Domain associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific.</p> <p>Root Complex Event Collectors provide support for the above-described functionality for RCiEPs.</p> <p>Default value of this bit is 0b.</p> | RW |
| 3 | <p>PME Interrupt Enable - When Set, this bit enables PME interrupt generation upon receipt of a PME Message as reflected in the PME Status bit (see § Table 7-32). A PME interrupt is also generated if the PME Status bit is Set when this bit is changed from Clear to Set (see § Section 5.3.3).</p> <p>Default value of this bit is 0b.</p> | RW |
| 4 | <p>Configuration RRS Software Visibility Enable - When Set, this bit enables the Root Port to indicate to software when Request Retry Status (RRS) Completion Status is received in response to a Configuration Request (see § Section 2.3.1).</p> <p>Root Ports that do not implement this capability must hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 5 | <p>No NFM Subtree Below This Root Port - When Clear, indicates that the RC must take ownership of non-UIO Non-Posted Requests passing peer-to-peer through the RC targeting this RP as the Egress Port. It is strongly recommended that system software Set this bit if it is determined that no NFM subtree(s) exist below this Root Port. RC implementations are strongly recommended to avoid taking ownership when not required to do so.</p> <p>Root Ports must handle the Clearing of this bit without disruption to Non-Posted Requests the RC has taken ownership of.</p> <p>Root Ports that do not implement this capability must hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW |

7.5.3.13 Root Capabilities Register (Offset 1Eh) §

The Root Capabilities Register identifies PCI Express Root Port specific capabilities. § Figure 7-34 details allocation of register fields in the Root Capabilities Register ; § Table 7-31 provides the respective bit definitions.

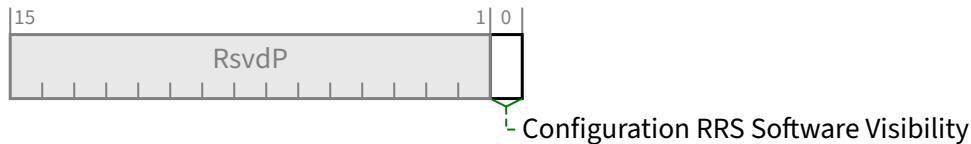


Figure 7-34 Root Capabilities Register §

Table 7-31 Root Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Configuration RRS Software Visibility - When Set, this bit indicates that the Root Port is capable of indicating to software when Request Retry Status (RRS) Completion Status is received in response to a Configuration Request (see § Section 2.3.1). | RO |

7.5.3.14 Root Status Register (Offset 20h) §

The Root Status Register provides information about PCI Express device specific parameters. § Figure 7-35 details allocation of register fields in the Root Status Register ; § Table 7-32 provides the respective bit definitions.

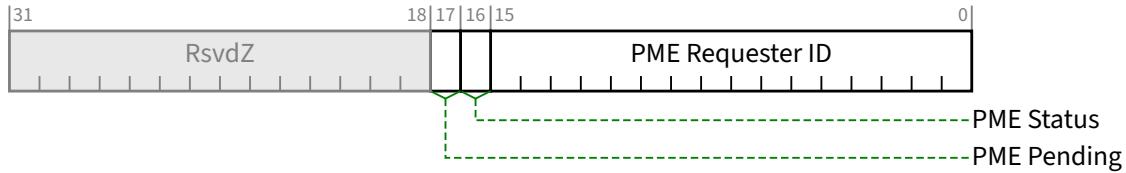


Figure 7-35 Root Status Register §

Table 7-32 Root Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PME Requester ID - This field indicates the PCI Requester ID of the last PME Requester. This field is only valid when the PME Status bit is Set. | RO |
| 16 | PME Status - This bit indicates that PME was asserted by the PME Requester indicated in the PME Requester ID field. Subsequent PMEs are kept pending until the status register is cleared by software by writing a 1b. Default value of this bit is 0b. | RW1C |
| 17 | PME Pending - This bit indicates that another PME is pending when the PME Status bit is Set. When the PME Status bit is cleared by software; the PME is delivered by hardware by setting the PME Status bit again and updating the PME Requester ID field appropriately. The PME Pending bit is cleared by hardware if no more PMEs are pending. | RO |

Base 6.4 vs Base 6.3

7.5.3.15 Device Capabilities 2 Register (Offset 24h) §

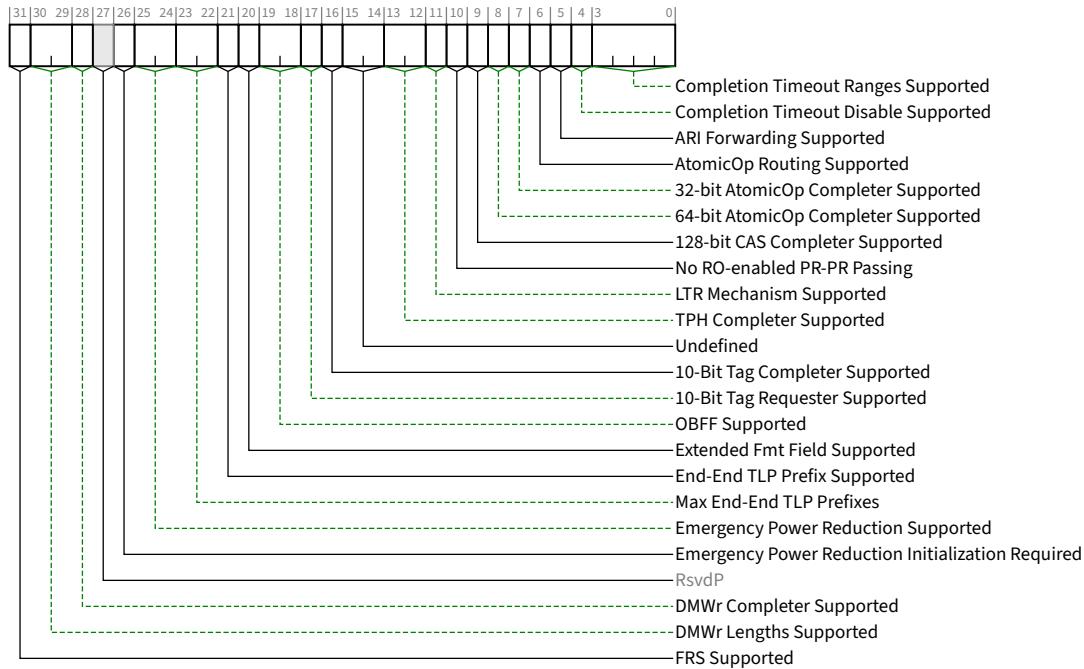


Figure 7-36 Device Capabilities 2 Register §

Table 7-33 Device Capabilities 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|--|
| 3:0 | <p>Completion Timeout Ranges Supported - This field indicates device Function support for the optional Completion Timeout programmability mechanism. This mechanism allows system software to modify the Completion Timeout Value .</p> <p>This field is applicable only to Root Ports, Endpoints that issue Non-Posted Requests N for UIO Requests on their own behalf, and PCI Express to PCI/PCI-X Bridges that take ownership of Non-Posted Requests issued on PCI Express. For all other Functions this field is Reserved and must be hardwired to 0000b.</p> <p>Four time value ranges are defined (A, B, C, D), each with two selectable sub-ranges (for which the time ranges are defined in the description of the Completion Timeout Value field in the Device Control 2 register):</p> <p>The value in this field indicates the timeout value ranges supported:</p> <ul style="list-style-type: none"> 0000b Completion Timeout programming not supported - See § Section 2.8 for requirements. 0001b Range A 0010b Range B 0011b Ranges A and B 0110b Ranges B and C 0111b Ranges A, B, and C 1110b Ranges B, C, and D | <p>HwInit</p> <div style="border: 1px solid red; padding: 5px; background-color: red; color: white; text-align: center;"> Errata: Base 6.3 B834Δ </div> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|--|
| | <p>1111b Ranges A, B, C, and D All other values are Reserved. For VFs, this field value must be identical to the associated PF's field value.</p> | |
| 4 | <p>Completion Timeout Disable Supported - A value of 1b indicates support for the Completion Timeout Disable mechanism. The Completion Timeout Disable mechanism is required for Endpoints that issue Non-Posted Requests ↑or UIO Requests on their own behalf and PCI Express to PCI/PCI-X Bridges that take ownership of Non-Posted Requests issued on PCI Express. For VFs, this bit value must be identical to the associated PF's bit value. This mechanism is optional for Root Ports. For all other Functions this field is Reserved and must be hardwired to 0b.</p> | RO <div style="background-color: red; color: white; padding: 2px 10px; border-radius: 10px; text-align: center;">Errata: Base 6.3 B834△◀▷</div> |
| 5 | <p>ARI Forwarding Supported - Applicable only to Switch Downstream Ports and Root Ports; must be 0b for other Function types. This bit must be set to 1b if a Switch Downstream Port or Root Port supports this optional capability. See § Section 6.13 for additional details.</p> | RO |
| 6 | <p>AtomicOp Routing Supported - Applicable only to Switch Upstream Ports, Switch Downstream Ports, and Root Ports; must be 0b for other Function types. This bit must be set to 1b if the Port supports this optional capability. See § Section 6.15 for additional details.</p> | RO |
| 7 | <p>32-bit AtomicOp Completer Supported - Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. Includes FetchAdd, Swap, and CAS AtomicOps. This bit must be set to 1b if the Function supports this optional capability. See § Section 6.15.3.1 for additional RC requirements. For VFs, this bit value must be identical to the associated PF's bit value.</p> | RO |
| 8 | <p>64-bit AtomicOp Completer Supported - Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. Includes FetchAdd, Swap, and CAS AtomicOps. This bit must be set to 1b if the Function supports this optional capability. See § Section 6.15.3.1 for additional RC requirements. For VFs, this bit value must be identical to the associated PF's bit value.</p> | RO |
| 9 | <p>128-bit CAS Completer Supported - Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. This bit must be set to 1b if the Function supports this optional capability. See § Section 6.15 for additional details. For VFs, this bit value must be identical to the associated PF's bit value.</p> | RO |
| 10 | <p>No RO-enabled PR-PR Passing - If this bit is Set, the routing element never carries out the passing permitted by ↑\$ Table 2-42 entry A2b that is associated with the Relaxed Ordering Attribute field being Set. ↑This bit only applies to ordering of TLPs for which the rules of § Section 2.4.1 apply. Errata: Base 6.3 B819△◀▷ This bit applies only for Switches and RCs that support peer-to-peer traffic between Root Ports. This bit applies only to Posted Requests being forwarded through the Switch or RC and does not apply to traffic originating or terminating within the Switch or RC itself. All Ports on a Switch or RC must report the same value for this bit. For all other functions, this bit must be 0b.</p> | HwInit |
| 11 | <p>LTR Mechanism Supported - A value of 1b indicates support for the optional Latency Tolerance Reporting (LTR) mechanism. Root Ports, Switches and Endpoints are permitted to implement this capability.</p> | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|--|
| | <p>For a Multi-Function Device associated with an Upstream Port, each Function must report the same value for this bit.</p> <p>For Bridges and other Functions that do not implement this capability, this bit must be hardwired to 0b.</p> | |
| 13:12 | <p>TPH Completer Supported - Value indicates Completer support for TPH or Extended TPH. Applicable only to Root Ports and Endpoints. For all other Functions, this field is Reserved.</p> <p>Defined Encodings are:</p> <ul style="list-style-type: none"> 00b TPH and Extended TPH Completer not supported. 01b TPH Completer supported; Extended TPH Completer not supported. 10b Reserved. 11b Both TPH and Extended TPH Completer supported. <p>See § Section 6.17 for details.</p> | RO |
| 15:14 | Undefined - formerly used for Lightweight Notification (LN), which is now deprecated | RO |
| 16 | <p>10-Bit Tag Completer Supported - If this bit is Set, the Function supports 10-Bit Tag Completer capability; otherwise, the Function does not. See § Section 2.2.6.2 .</p> <p>For VFs, this bit value must be identical to the associated PF's bit value.</p> | HwInit |
| 17 | <p>10-Bit Tag Requester Supported - If this bit is Set, the Function supports 10-Bit Tag Requester ↓↓capability; ↓↑capability for non-UIO Requests; ↑ otherwise, the Function does not. ↑↑This bit is not applicable to a Requester when generating UIO Requests.↑</p> <p>This bit must not be Set if the <u>10-Bit Tag Completer Supported</u> bit is Clear.</p> <p>If the Function is an RCiEP, this bit must be Clear if the RC does not support 10-Bit Tag Completer capability for Requests coming from this RCiEP.</p> <p>For VFs, this bit value must equal the <u>VF 10-Bit Tag Requester Supported</u> bit value in the SR-IOV Capabilities Register .</p> <p>Note that 10-Bit Tag field generation must be enabled by the <u>10-Bit Tag Requester Enable</u> bit in the <u>Device Control 2 Register</u> of the Requester Function before 10-Bit Tags can be generated by the Requester. See § Section 2.2.6.2 .</p> | Errata: Base 6.3 B807△◀▷ |
| 19:18 | <p>OBFF Supported - This field indicates if OBFF is supported and, if so, what signaling mechanism is used.</p> <ul style="list-style-type: none"> 00b OBFF Not Supported 01b OBFF supported using Message signaling only 10b OBFF supported using WAKE# signaling only 11b OBFF supported using WAKE# and Message signaling <p>The value reported in this field must indicate support for WAKE# signaling only if:</p> <ul style="list-style-type: none"> • for a Downstream Port, driving the WAKE# signal for OBFF is supported and the connector or component connected Downstream is known to receive that same WAKE# signal • for an Upstream Port, receiving the WAKE# signal for OBFF is supported and, if the component is on an add-in-card, that the component is connected to the WAKE# signal on the connector. <p>Root Ports, Switch Ports, and Endpoints are permitted to implement this capability.</p> <p>For a Multi-Function Device associated with an Upstream Port, each Function must report the same value for this field.</p> <p>For Bridges and Ports that do not implement this capability, this field must be hardwired to 00b.</p> | HwInit |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 20 | <p>Extended Fmt Field Supported - If Set, the Function supports the 3-bit definition of the Fmt field when operating in Non-Flit Mode. If Clear, the Function supports a 2-bit definition of the Fmt field. See § Section 2.2.</p> <p>Must be Set for Functions that support End-End TLP Prefixes (NFM) or OHC-E (FM). All Functions in an Upstream Port must have the same value for this bit. Each Downstream Port of a component may have a different value for this bit.</p> <p><i>MUST</i>@FLIT be Set.</p> | RO |
| 21 | <p>End-End TLP Prefix Supported - Indicates whether End-End TLP Prefix support (NFM) / OHC-E (FM) is offered by a Function. Values are:</p> <ul style="list-style-type: none"> 0b No Support 1b Support is provided to receive TLPs containing End-End TLP Prefixes (NFM) and optionally OHC-E (FM). <p>All Ports of a Switch must have the same value for this bit.</p> <p>The definition of this bit is ambiguous for designs that choose only to support End-End TLP Prefixes in NFM and do not support OHC-E in FM. This bit is static and does not change value with current link operation (FM vs. NFM). Software cannot rely on this bit to infer if OHC-E is supported by a Function.</p> <p>The definition of this bit is also ambiguous for RPs that choose to support End-End TLP Prefixes (NFM) / OHC-E (FM) as a terminus only without forwarding. Software cannot rely on this bit to infer if End-End TLP Prefix forwarding / OHC-E forwarding is supported in a RP or not.</p> | HwInit |
| 23:22 | <p>Max End-End TLP Prefixes - Indicates the maximum number of End-End TLP Prefixes supported by this Function (NFM) or the maximum size of OHC-E supported (FM). See § Section 2.2.10.4 for important details. Values are:</p> <ul style="list-style-type: none"> 01b 1 End-End TLP Prefix / OHC-E1 10b 2 End-End TLP Prefixes / OHC-E2 11b 3 End-End TLP Prefixes / OHC-E4 00b 4 End-End TLP Prefixes / OHC-E4 <p>If End-End TLP Prefix Supported is Clear, this field is RsvdP.</p> <p>Different Root Ports that have the End-End TLP Prefix Supported bit Set are permitted to report different values for this field.</p> <p>For Switches where End-End TLP Prefix Supported is Set, this field must be 00b indicating support for up to four End-End TLP Prefixes.</p> <p>The definition of this bit is ambiguous for designs that only chose to support End-End TLP Prefixes in NFM and do not support OHC-E in FM. This bit is static and does not change value based on current link operation (FM vs. NFM). Refer to § Section 2.2.11 for how hardware must handle received TLPs with OHC-E in the scenario that they don't support it.</p> | HwInit |
| 25:24 | <p>Emergency Power Reduction Supported - Indicates support level of the optional Emergency Power Reduction State feature. A Function can enter Emergency Power Reduction State autonomously, or based on one of two mechanisms defined by the associated Form Factor Specification. Functions that are in the Emergency Power Reduction State consume less power. The Emergency Power Reduction mechanism permits a chassis to request add-in cards to rapidly enter Emergency Power Reduction State without involving system software. See § Section 6.24 for additional details.</p> <p>Values are:</p> <ul style="list-style-type: none"> 00b Emergency Power Reduction State not supported 01b Emergency Power Reduction State is supported and is triggered by Device Specific mechanism(s) | HwInit |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| | <p>10b Emergency Power Reduction State is supported and is triggered either by the mechanism defined in the corresponding Form Factor specification or by Device Specific mechanism(s)</p> <p>11b Reserved</p> <p>This field is RsvdP in Functions that are not associated with an Upstream Port.</p> <p>For Multi-Function Devices associated with an Upstream Port, all Functions that report a non-Zero value for this field, must report the same non-Zero value for this field.</p> <p>For VFs, this field value must be identical to the associated PF's field value.</p> <p>Default value is 00b.</p> <p>After reset, once this field returns a non-Zero value, it must continue to return the same non-Zero value, until the next reset.</p> | |
| 26 | <p>Emergency Power Reduction Initialization Required - If Set, the Function requires complete or partial initialization upon exit from the Emergency Power Reduction State. If Clear, the Function requires no software intervention to return to normal operation upon exit from the Emergency Power Reduction State. See § Section 6.24 for additional details.</p> <p>For Multi-Function Devices associated with an Upstream Port, all Functions must report the same value for this bit.</p> <p>For VFs, this bit value must be identical to the associated PF's bit value.</p> <p>This bit is RsvdP in Functions that are not associated with an Upstream Port.</p> <p>Default value is 0b.</p> <p>After reset, when this field returns a non-Zero value, it must continue to return the same non-Zero value.</p> | HwInit |
| 28 | <p>DMWr Completer Supported – Applicable to Functions with Memory Space BARs as well as all Root Ports; This bit must be Set if the Function can serve as a DMWr Completer. See § Section 6.32 for additional details.</p> | HwInit |
| 30:29 | <p>DMWr Lengths Supported – Applicable to Functions with either the DMWr Request Routing Supported bit Set or the DMWR Completer Supported bit Set (or both). This field indicates the largest DMWr TLP that this Function can receive.</p> <p>Defined Encodings are:</p> <ul style="list-style-type: none"> 00b DMWr TLPs up to 64 bytes are supported 01b DMWr TLPs up to 128 bytes are supported 10b Reserved 11b Reserved <p>When applicable, all Functions in a Multi-Function Device associated with an Upstream Port must report the same value in this field.</p> <p>This field is RsvdP if both DMWr Completer Supported and DMWr Request Routing Supported are Clear.</p> | HwInit / RsvdP |
| 31 | <p>FRS Supported - When Set, indicates support for the optional Function Readiness Status (FRS) capability.</p> <p>Must be Set for all Functions that support generation or reception capabilities of FRS Messages.</p> <p>Must not be Set by Switch Functions that do not generate FRS Messages on their own behalf.</p> | HwInit |

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: USE OF THE NO RO-ENABLED PR-PR PASSING BIT §

The No RO-enabled PR-PR Passing bit allows platforms to utilize PCI Express switching elements on the path between a requester and completer for requesters that could benefit from a slightly less relaxed ordering model. An example is a device that cannot ensure that multiple overlapping posted writes to the same address are outstanding at the same time. The method by which such a device is enabled to utilize this mode is beyond the scope of this specification.

7.5.3.16 Device Control 2 Register (Offset 28h) §

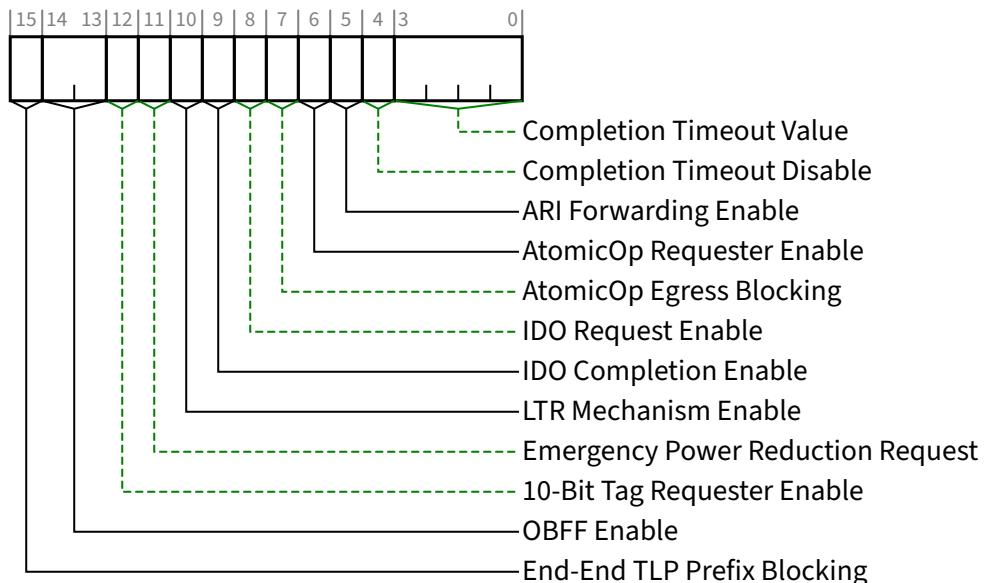


Figure 7-37 Device Control 2 Register §

Table 7-34 Device Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|--|
| 3:0 | <p>Completion Timeout Value - In device Functions that support Completion Timeout programmability, this field allows system software to modify the Completion Timeout Value.</p> <p>This field is applicable to Root Ports, Endpoints that issue Non-Posted Requests ↑↑or UIO Requests↑↑ on their own behalf, and PCI Express to PCI/PCI-X Bridges that take ownership of Non-Posted Requests issued on PCI Express. For VFs, the associated PF's value applies, and this field must be RsvdP. For all other Functions, this field must be hardwired to Zero.</p> <p>A Function that does not support this optional capability must hardwire this field to 0000b. Functions that support Completion Timeout programmability must support the values given below</p> | RW VF RsvdP <div style="border: 1px solid red; padding: 2px; margin-top: 10px;"> Errata: Base 6.3 B834△↔ </div> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| | <p>corresponding to the programmability ranges indicated in the <u>Completion Timeout Ranges Supported</u> field.</p> <p>Defined encodings:</p> <ul style="list-style-type: none"> 0000b Default range: 50 µs to 50 ms; the Function <i>MUST</i>@FLIT implement a timeout value in the range 40 ms to 50 ms. <p>For Functions that do not support Flit Mode, it is strongly recommended that the Completion Timeout mechanism not expire in less than 10 ms.</p> <p>Values available if Range A is supported:</p> <ul style="list-style-type: none"> 0001b 50 µs to 100 µs 0010b 1 ms to 10 ms <p>Values available if Range B is supported:</p> <ul style="list-style-type: none"> 0101b 16 ms to 55 ms ; <i>MUST</i>@FLIT 40 ms to 55 ms 0110b 65 ms to 210 ms <p>Values available if Range C is supported:</p> <ul style="list-style-type: none"> 1001b 260 ms to 900 ms 1010b 1 s to 3.5 s <p>Values available if the Range D is supported:</p> <ul style="list-style-type: none"> 1101b 4 s to 13 s 1110b 17 s to 64 s <p>Values not defined above are Reserved.</p> <p>Software is permitted to change the value in this field at any time. For Requests already pending when the Completion Timeout Value is changed, hardware is permitted to use either the new or the old value for the outstanding Requests, and is permitted to base the start time for each Request either on when this value was changed or on when each request was issued.</p> <p>The default value for this field is 0000b.</p> | |
| 4 | <p>Completion Timeout Disable - When Set, this bit disables the Completion Timeout mechanism.</p> <p>For non-VFs, this bit is required for all Functions that support the Completion Timeout Disable capability. For VFs, the associated PF's value applies, and this field must be RsvdP . Otherwise, Functions that do not support this optional capability are permitted to hardwire this bit to Zero .</p> <p>Software is permitted to Set or Clear this bit at any time. When Set, the Completion Timeout detection mechanism is disabled. If there are outstanding Requests when the bit is cleared, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding Requests. If this is done, it is permitted to base the start time for each Request on either the time this bit was cleared or the time each Request was issued.</p> <p>The default value for this bit is 0b.</p> | RW VF RsvdP |
| 5 | <p>ARI Forwarding Enable - When set, the Downstream Port disables its traditional Device Number field being 0 enforcement when turning a Type 1 Configuration Request into a Type 0 Configuration Request, permitting access to Extended Functions in an ARI Device immediately below the Port. See § Section 6.13 .</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the <u>ARI Forwarding Supported</u> bit is 0b.</p> <p>This bit is not applicable and Reserved for Upstream Ports.</p> | RW / RsvdP |
| 6 | <p>AtomicOp Requester Enable - Applicable only to Endpoints and Root Ports; must be hardwired to 0b for other Function types. For Endpoints, the Function is allowed to initiate AtomicOp Requests only if this bit and the Bus Master Enable bit in the Command register are both Set. For Root Ports, the CPU is</p> | RW VF RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------|
| | <p>allowed to initiate AtomicOp Requests on this Link only when this bit is Set and <u>AtomicOp Egress Blocking</u> is Clear (see § Section 6.15).</p> <p>For VFs, the associated PF's value applies, and this bit must be RsvdP . For non-VFs, this bit is required to be RW if the Endpoint or Root Port is capable of initiating AtomicOp Requests, but otherwise is permitted to be hardwired to Zero .</p> <p>This bit does not serve as a capability bit. This bit is permitted to be RW even if no AtomicOp Requester capabilities are supported by the Endpoint or Root Port.</p> <p>Default value of this bit is 0b.</p> | |
| 7 | <p>AtomicOp Egress Blocking - Applicable and mandatory for Switch Upstream Ports, Switch Downstream Ports, and Root Ports that implement AtomicOp routing capability; otherwise must be hardwired to 0b.</p> <p>When this bit is Set, AtomicOp Requests that target going out this Egress Port must be blocked. See § Section 6.15.2 .</p> <p>Default value of this bit is 0b.</p> | RW |
| 8 | <p>IDO Request Enable - If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attr[2]) of Requests it initiates (see § Section 2.2.6.3 and § Section 2.4).</p> <p>Endpoints, including RC Integrated Endpoints, and Root Ports are permitted to implement this capability.</p> <p>For VFs, the associated PF's value applies, and this bit must be RsvdP . Otherwise, a Function is permitted to hardwire this bit to Zero if it never sets the IDO attribute in Requests.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP |
| 9 | <p>IDO Completion Enable - If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attr[2]) of Completions it returns (see § Section 2.2.6.3 and § Section 2.4).</p> <p>Endpoints, including RC Integrated Endpoints, and Root Ports are permitted to implement this capability.</p> <p>For VFs, the associated PF's value applies, and this bit must be RsvdP . Otherwise, a Function is permitted to hardwire this bit to Zero if it never sets the IDO attribute in Completions.</p> <p>Default value of this bit is 0b.</p> | RW VF RsvdP |
| 10 | <p>LTR Mechanism Enable - When Set to 1b, this bit enables Upstream Ports to send LTR messages and Downstream Ports to process LTR Messages.</p> <p>For a Multi-Function Device associated with an Upstream Port of a device that implements LTR, the bit in Function 0 is RW , and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is RsvdP .</p> <p>Functions that do not implement the LTR mechanism are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> <p>For Downstream Ports, this bit must be reset to the default value if the Port goes to DL_Down status.</p> | RW / RsvdP |
| 11 | <p>Emergency Power Reduction Request - If Set, all Functions in the component that support Emergency Power Reduction State must enter the Emergency Power Reduction State . If Clear these Functions must exit the Emergency Power Reduction State if no other reasons exist to preclude exiting this state. See § Section 6.24 for additional details.</p> <p>This bit is implemented in the lowest numbered (non-VF) Function associated with an Upstream Port that has a non- Zero value in the Emergency Power Reduction Supported field. This bit is RsvdP in all other Functions, including VFs.</p> <p>Default is 0b.</p> | RW / RsvdP VF RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|--|
| 12 | <p>10-Bit Tag Requester Enable - This bit, in combination with the Extended Tag Field Enable bit and the 14-Bit Tag Requester Enable bit, determines how many Tag field bits a Requester is permitted to use. When this bit is Set, the Requester is permitted to use 10-Bit Tags for non-UIO Requests. See § Section 2.2.6.2 for complete details.</p> <p>If software changes the value of this bit while the Function has outstanding Non-Posted Requests, the result is undefined.</p> <p>For VFs, the value in the VF 10-Bit Tag Requester Enable bit in the associated PF's SR-IOV Control Register applies, and this bit must be RsvdP.</p> <p>Non-VF Functions that do not implement 10-Bit Tag Requester capability are permitted to hardwire this bit to Zero.</p> <p>Default value of this bit is 0b.</p> | <p>RW VF RsvdP</p> <p>Errata: Base 6.3 B807△↔</p> |
| 14:13 | <p>OBFF Enable - This field enables the OBFF mechanism and selects the signaling method.</p> <ul style="list-style-type: none"> 00b Disabled 01b Enabled using Message signaling [Variation A] 10b Enabled using Message signaling [Variation B] 11b Enabled using WAKE# signaling <p>See § Section 6.19 for an explanation of the above encodings.</p> <p>This field is required for all Ports that support the OBFF Capability.</p> <p>For a Multi-Function Device associated with an Upstream Port of a Device that implements OBFF, the field in Function 0 is of type RW, and only Function 0 controls the Component's behavior. In all other Functions of that Device, this field is of type RsvdP.</p> <p>Ports that do not implement OBFF are permitted to hardwire this field to 00b.</p> <p>Default value of this field is 00b.</p> | <p>RW / RsvdP (see description)</p> |
| 15 | <p>End-End TLP Prefix Blocking - Controls whether the routing function is permitted to forward TLPs containing an End-End TLP Prefix (NFM) / OHC-E (FM). Values are:</p> <ul style="list-style-type: none"> 0b Forwarding Enabled - Function is permitted to send TLPs with End-End TLP Prefixes (NFM) or OHC-E (FM). 1b Forwarding Blocked - Function is not permitted to send TLPs with End-End TLP Prefixes (NFM) or OHC-E (FM). <p>This bit affects TLPs that exit the Switch/Root Complex using the associated Port. It does not affect TLPs forwarded internally within the Switch/Root Complex. It does not affect TLPs that enter through the associated Port, that originate in the associated Port or originate in a Root Complex Integrated Device integrated with the associated Port. As described in § Section 2.2.10.4, blocked TLPs are reported by the TLP Prefix Blocked Error.</p> <p>The default value of this bit is 0b.</p> <p>This bit is hardwired to 1b in Root Ports that support End-End TLP Prefixes/ OHC-E but do not support forwarding of End-End TLP Prefixes/ OHC-E .</p> <p>This bit is applicable to Root Ports and Switch Ports where the End-End TLP Prefix Supported bit is Set or where the associated OHC-E Support is 001b, 010b, 011b or 100b. This bit is not applicable and is RsvdP in all other cases.</p> | <p>RW (see description)</p> |

7.5.3.17 Device Status 2 Register (Offset 2Ah) §

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdZ.

7.5.3.18 Link Capabilities 2 Register (Offset 2Ch) §

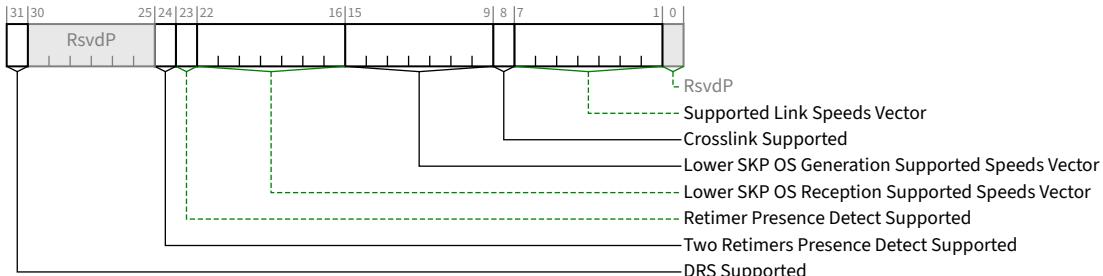


Figure 7-38 Link Capabilities 2 Register §

Table 7-35 Link Capabilities 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 7:1 | <p>Supported Link Speeds Vector - This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. See § Section 8.2.1 for further requirements.</p> <p>Bit definitions within this field are:</p> <ul style="list-style-type: none"> Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bit 4 32.0 GT/s Bit 5 64.0 GT/s Bit 6 RsvdP <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | HwInit / RsvdP |
| 8 | <p>Crosslink Supported - When set to 1b, this bit indicates that the associated Port supports crosslinks (see § Section 4.2.7.3.1). When set to 0b on a Port that supports Link speeds of 8.0 GT/s or higher, this bit indicates that the associated Port does not support crosslinks. When set to 0b on a Port that only supports Link speeds of 2.5 GT/s or 5.0 GT/s, this bit provides no information regarding the Port's level of crosslink support.</p> <p>It is recommended that this bit be Set in any Port that supports crosslinks even though doing so is only required for Ports that also support operating at 8.0 GT/s or higher Link speeds.</p> <p>Note: Software should use this bit when referencing fields whose definition depends on whether or not the Port supports crosslinks (see § Section 7.7.3.4).</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------|
| 15:9 | <p>Lower SKP OS Generation Supported Speeds Vector - If this field is non-Zero, it indicates that the Port, when operating at the indicated speed(s) supports SRIS and also supports software control of the SKP Ordered Set transmission scheduling rate.</p> <p>Bit definitions within this field are:</p> <ul style="list-style-type: none"> Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bit 4 32.0 GT/s Bit 5 64.0 GT/s Bit 6 RsvdP <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> <p>Behavior is undefined if a bit is Set in this field and the corresponding bit is not Set in the Supported Link Speeds Vector.</p> | HwInit / RsvdP |
| 22:16 | <p>Lower SKP OS Reception Supported Speeds Vector - If this field is non-Zero, it indicates that the Port, when operating at the indicated speed(s) supports SRIS and also supports receiving SKP OS at the rate defined for SRNS while running in SRIS.</p> <p>Bit definitions within this field are:</p> <ul style="list-style-type: none"> Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bit 4 32.0 GT/s Bit 5 64.0 GT/s Bit 6 RsvdP <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> <p>Behavior is undefined if a bit is Set in this field and the corresponding bit is not Set in the Supported Link Speeds Vector.</p> | HwInit / RsvdP |
| 23 | <p>Retimer Presence Detect Supported - When set to 1b, this bit indicates that the associated Port supports detection and reporting of Retimer presence.</p> <p>This bit <i>MUST</i>@FLIT be Set.</p> <p>This bit must be set to 1b in a Port when the Supported Link Speeds Vector of the Link Capabilities 2 Register indicates support for a Link speed of 16.0 GT/s or higher.</p> <p>It is permitted to be set to 1b regardless of the supported Link speeds.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> | HwInit / RsvdP |
| 24 | <p>Two Retimers Presence Detect Supported - When set to 1b, this bit indicates that the associated Port supports detection and reporting of two Retimers presence.</p> <p>This bit <i>MUST</i>@FLIT be Set.</p> | HwInit / RsvdP |

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 25 | <p>This bit must be set to 1b in a Port when the Supported Link Speeds Vector of the Link Capabilities 2 Register indicates support for a Link speed of 16.0 GT/s or higher.</p> <p>It is permitted to be set to 1b regardless of the supported Link speeds if the Retimer Presence Detect Supported bit is also set to 1b.</p> <p>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.</p> <p>ECN: Base 6.3 Optical</p> <p><i>Optical Retimer Presence Detect Supported – When set to 1b, this bit indicates that the associated Port supports detection and reporting of Optical Retimer Presence.</i></p> <p><i>Multi-Function Devices associated with an Upstream Port must report the same value in this field for all functions.</i></p> | |
| 31 | <p>DRS Supported - When Set, indicates support for the optional Device Readiness Status (DRS) capability.</p> <p>Must be Set in Downstream Ports that support DRS.</p> <p>Must be Set in Downstream Ports that support FRS.</p> <p>For Upstream Ports that support DRS, this bit <i>MUST</i> be Set in Function 0. For all other Functions associated with an Upstream Port, this bit must be Clear.¹⁸¹</p> <p>Must be Clear in Functions that are not associated with a Port.</p> <p>RsvdP in all other Functions.</p> | HwInit / RsvdP |

IMPLEMENTATION NOTE: SOFTWARE MANAGEMENT OF LINK SPEEDS WITH EARLIER HARDWARE §

Hardware components compliant to versions prior to [PCIe-3.0] either did not implement the Link Capabilities 2 Register, or the register was Reserved.

For software to determine the supported Link speeds for components where the Link Capabilities 2 Register is either not implemented, or the value of its Supported Link Speeds Vector is 0000000b, software can read bits 3:0 of the Link Capabilities Register (now defined to be the Max Link Speed field), and interpret the value as follows:

0001b

2.5 GT/s Link speed supported

0010b

5.0 GT/s and 2.5 GT/s Link speeds supported

For such components, the encoding of the values for the Current Link Speed field (in the Link Status Register) and Target Link Speed field (in the Link Control 2 Register) is the same as above.

¹⁸¹. It is expressly permitted for Upstream Ports to send DRS Messages even when the DRS Supported bit is Clear.

IMPLEMENTATION NOTE: SOFTWARE MANAGEMENT OF LINK SPEEDS WITH FUTURE HARDWARE §

It is strongly encouraged that software primarily utilize the Supported Link Speeds Vector instead of the Max Link Speed field, so that software can determine the exact set of supported speeds on current and future hardware. This can avoid software being confused if a future specification defines Links that do not require support for all slower speeds.

7.5.3.19 Link Control 2 Register (Offset 30h) §

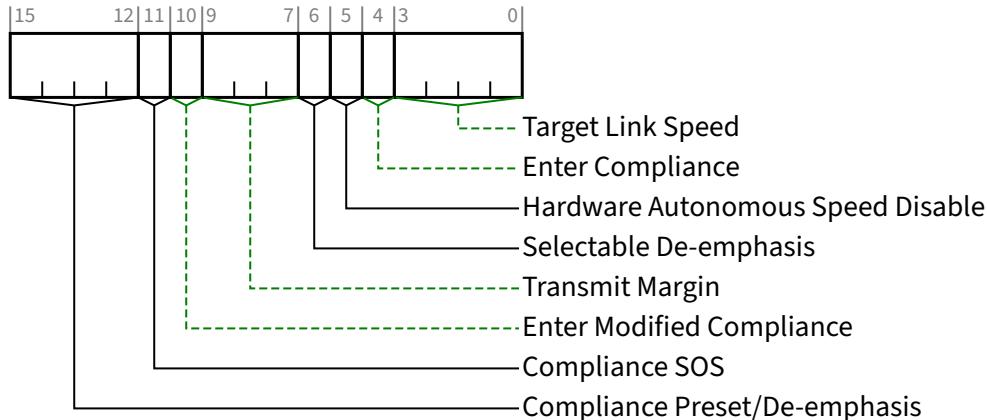


Figure 7-39 Link Control 2 Register §

Table 7-36 Link Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------------------|
| 3:0 | <p>Target Link Speed - For Downstream Ports, this field sets an upper limit on Link operational speed by restricting the values advertised by the Upstream component in its training sequences.</p> <p>The encoded value specifies a Bit Location in the Supported Link Speeds Vector (in the Link Capabilities 2 Register) that corresponds to the desired target Link speed.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 | RWS / RsvdP (see description) |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | |
|--------------|--|---|---------|-----------|-------|---------------|
| | <p>Others All other encodings are Reserved.</p> <p>If a value is written to this field that does not correspond to a supported speed (as indicated by the Supported Link Speeds Vector), the result is undefined.</p> <p>If either of the <u>Enter Compliance</u> or <u>Enter Modified Compliance</u> bits are implemented, then this field must also be implemented.</p> <p>The default value of this field is the highest Link speed supported by the component (as reported in the Max Link Speed field of the <u>Link Capabilities Register</u>) unless the corresponding platform/form factor requires a different default value.</p> <p>For both Upstream and Downstream Ports, this field is used to set the target compliance mode speed when software is using the <u>Enter Compliance</u> bit to force a Link into compliance mode.</p> <p>For Upstream Ports, if the <u>Enter Compliance</u> bit is Clear, this field is permitted to have no effect.</p> <p>For a Multi-Function Device associated with an Upstream Port, the field in Function 0 is of type <u>RWS</u>, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this field is of type <u>RsvdP</u>.</p> <p>Components that support only the 2.5 GT/s speed are permitted to hardwire this field to 0000b.</p> | | | | | |
| 4 | <p>Enter Compliance - Software is permitted to force a Link to enter Compliance mode (at the speed indicated in the Target Link Speed field and the de-emphasis/preset level indicated by the Compliance Preset/De-emphasis bit) by setting this bit to 1b in both components on a Link and then initiating a Hot Reset on the Link.</p> <p>Default value of this bit following Fundamental Reset is 0b.</p> <p>For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type <u>RWS</u>, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type <u>RsvdP</u>.</p> <p>Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b.</p> <p>This bit is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this bit only during debug or compliance testing. In all other cases, the system must ensure that this bit is Set to the default value.</p> | <u>RWS</u> / <u>RsvdP</u> (see description) | | | | |
| 5 | <p>Hardware Autonomous Speed Disable - When Set, this bit disables hardware from changing the Link speed for device-specific reasons other than attempting to correct unreliable Link operation by reducing Link speed. Initial transition to the highest supported common link speed is not blocked by this bit.</p> <p>For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type <u>RWS</u>, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type <u>RsvdP</u>.</p> <p>Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | <u>RWS</u> / <u>RsvdP</u> (see description) | | | | |
| 6 | <p>Selectable De-emphasis - When the Link is operating at 5.0 GT/s speed, this bit is used to control the transmit de-emphasis of the link in specific situations. See § Section 4.2.7 for detailed usage information.</p> <p>Encodings:</p> <table style="margin-left: 20px;"> <tr> <td>1b</td> <td>-3.5 dB</td> </tr> <tr> <td>0b</td> <td>-6 dB</td> </tr> </table> <p>When the Link is not operating at 5.0 GT/s speed, the setting of this bit has no effect. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b.</p> <p>This bit is not applicable and Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches.</p> | 1b | -3.5 dB | 0b | -6 dB | <u>HwInit</u> |
| 1b | -3.5 dB | | | | | |
| 0b | -6 dB | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|-------------------------------|
| 9:7 | <p>Transmit Margin - This field controls the value of the non-deemphasized voltage level at the Transmitter pins. This field is reset to 000b on entry to the LTSSM Polling.Configuration substate (see § Chapter 4. for details of how the Transmitter voltage level is determined in various states).</p> <p>Encodings:</p> <ul style="list-style-type: none"> 000b Normal operating range 001b-111b As defined in § Section 8.3.4 not all encodings are required to be implemented. <p>For a Multi-Function Device associated with an Upstream Port, the field in Function 0 is of type RWS , and only Function 0 controls the component's Link behavior. In all other Functions of that device, this field is of type RsvdP .</p> <p>Default value of this field is 000b.</p> <p>Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 000b.</p> <p>This field is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this field only during debug or compliance testing. In all other cases, the system must ensure that this field is set to the default value.</p> | RWS / RsvdP (see description) |
| 10 | <p>Enter Modified Compliance - When this bit is Set to 1b, the device transmits Modified Compliance Pattern if the LTSSM enters Polling.Compliance substate.</p> <p>Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b.</p> <p>For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RWS , and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP .</p> <p>Default value of this bit is 0b.</p> <p>This bit is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this bit only during debug or compliance testing. In all other cases, the system must ensure that this bit is Set to the default value.</p> | RWS / RsvdP (see description) |
| 11 | <p>Compliance SOS - When set to 1b, the LTSSM is required to send SKP Ordered Sets between sequences when sending the Compliance Pattern or Modified Compliance Pattern.</p> <p>For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RWS , and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP .</p> <p>The default value of this bit is 0b.</p> <p>This bit is applicable when the Link is operating at 2.5 GT/s or 5.0 GT/s data rates only.</p> <p>Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b.</p> | RWS / RsvdP (see description) |
| 15:12 | <p>Compliance Preset/De-emphasis -</p> <p>For 8.0 GT/s and higher Data Rate: This field sets the Transmitter Preset in Polling.Compliance state if the entry occurred due to the Enter Compliance bit being 1b. The encodings are defined in § Section 4.2.4.2 . Results are undefined if a reserved preset encoding is used when entering Polling.Compliance in this way.</p> <p>For 5.0 GT/s Data Rate: This field sets the de-emphasis level in Polling.Compliance state if the entry occurred due to the Enter Compliance bit being 1b.</p> <p>Defined Encodings are:</p> <ul style="list-style-type: none"> 0001b -3.5 dB 0000b -6 dB <p>When the Link is operating at 2.5 GT/s, the setting of this field has no effect. Components that support only 2.5 GT/s speed are permitted to hardwire this field to 0000b.</p> | RWS / RsvdP (see description) |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>For a Multi-Function Device associated with an Upstream Port, the field in Function 0 is of type RWS , and only Function 0 controls the component's Link behavior. In all other Functions of that device, this field is of type RsvdP .</p> <p>The default value of this field is 0000b.</p> <p>This field is intended for debug and compliance testing purposes. System firmware and software is allowed to modify this field only during debug or compliance testing. In all other cases, the system must ensure that this field is set to the default value.</p> | |

IMPLEMENTATION NOTE: SELECTABLE DE-EMPHASIS USAGE §

Selectable De-emphasis setting is applicable only to Root Ports and Switch Downstream Ports. The De-emphasis setting is implementation specific and depends on the platform or enclosure in which the Root Port or the Switch Downstream Port is located. System firmware or hardware strapping is used to configure the Selectable De-emphasis value. In cases where system firmware cannot be used to set the de-emphasis value (for example, a hot plugged Switch), hardware strapping must be used to set the de-emphasis value.

7.5.3.20 Link Status 2 Register (Offset 32h) §

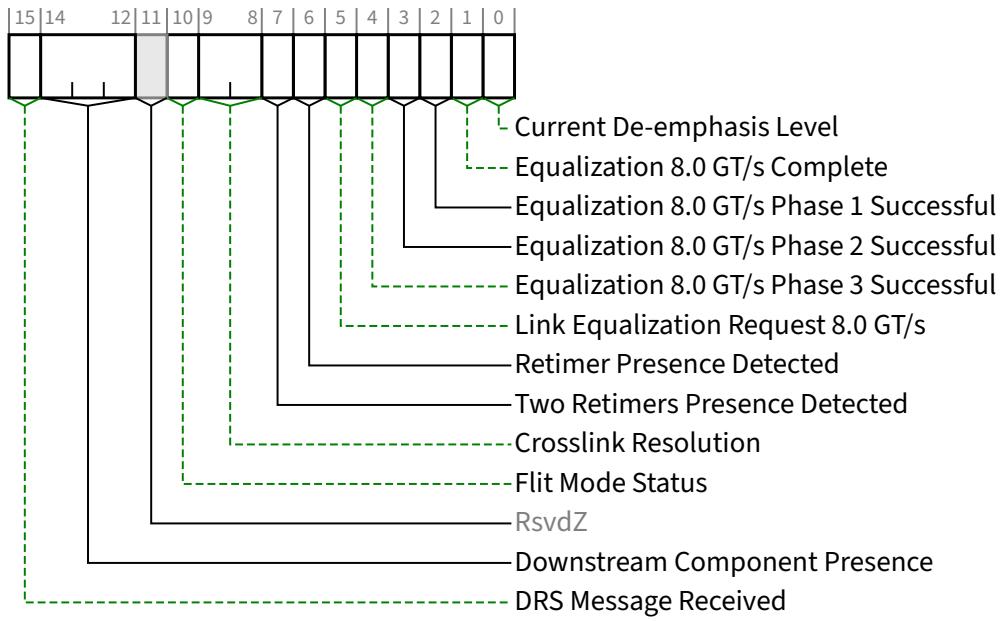


Figure 7-40 Link Status 2 Register §

Table 7-37 Link Status 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------------|
| 0 | <p>Current De-emphasis Level - When the Link is operating at 5.0 GT/s speed, this bit reflects the level of de-emphasis.</p> <p>Encodings:</p> <ul style="list-style-type: none"> 1b -3.5 dB 0b -6 dB <p>The value in this bit is undefined when the Link is not operating at 5.0 GT/s speed.</p> <p>For VFs, the associated PF's value applies, and this field must be RsvdZ. Otherwise, components that support only the 2.5 GT/s speed are permitted to hardwire this bit to Zero.</p> <p>For components that support speeds greater than 2.5 GT/s, Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions of the Port.</p> | <u>RO</u> <u>VF RsvdZ</u> |
| 1 | <p>Equalization 8.0 GT/s Complete - When set to 1b, this bit indicates that the Transmitter Equalization procedure at the 8.0 GT/s data rate has completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.</p> | <u>ROS</u> |
| 2 | <p>Equalization 8.0 GT/s Phase 1 Successful - When set to 1b, this bit indicates that Phase 1 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.</p> | <u>ROS</u> |
| 3 | <p>Equalization 8.0 GT/s Phase 2 Successful - When set to 1b, this bit indicates that Phase 2 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.</p> | <u>ROS</u> |
| 4 | <p>Equalization 8.0 GT/s Phase 3 Successful - When set to 1b, this bit indicates that Phase 3 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.</p> | <u>ROS</u> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|-------------|
| 5 | <p>Link Equalization Request 8.0 GT/s - This bit is Set by hardware to request the 8.0 GT/s Link equalization process to be performed on the Link. Refer to § <u>Section 4.2.4</u> and § <u>Section 4.2.7.4.2</u> for details.</p> <p>The default value of this bit is 0b.</p> <p>For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.</p> | RW1CS |
| 6 | <p>Retimer Presence Detected - When set to 1b, this bit indicates that a Retimer was present during the most recent Link negotiation. Refer to § <u>Section 4.2.7.3.5.1</u> for details.</p> <p>The default value of this bit is 0b.</p> <p>This bit is required for Ports that have the <u>Retimer Presence Detect Supported</u> bit of the <u>Link Capabilities 2 Register</u> set to 1b.</p> <p>Ports that have the <u>Retimer Presence Detect Supported</u> bit set to 0b are permitted to hardwire this bit to 0b.</p> <p>For Multi-Function Devices associated with an Upstream Port, this bit must be implemented in Function 0 and is RsvdZ in all other Functions.</p> | ROS / RsvdZ |
| 7 | <p>Two Retimers Presence Detected - When set to 1b, this bit indicates that two Retimers were present during the most recent Link negotiation. Refer to § <u>Section 4.2.7.3.5.1</u> for details.</p> <p>The default value of this bit is 0b.</p> <p>This bit is required for Ports that have the <u>Two Retimers Presence Detect Supported</u> bit of the <u>Link Capabilities 2 Register</u> set to 1b.</p> <p>Ports that have the <u>Two Retimers Presence Detect Supported</u> bit set to 0b are permitted to hardwire this bit to 0b.</p> <p>For Multi-Function Devices associated with an Upstream Port, this bit must be implemented in Function 0 and RsvdZ in all other Functions.</p> | ROS / RsvdZ |
| 9:8 | <p>Crosslink Resolution - This field indicates the state of the Crosslink negotiation. It must be implemented if <u>Crosslink Supported</u> is Set and the Port supports 16.0 GT/s or higher data rate. It is permitted to be implemented in all other Ports. If <u>Crosslink Supported</u> is Clear, this field may be hardwired to 01b or 10b.</p> <p>Encoding is:</p> <ul style="list-style-type: none"> 00b Crosslink Resolution is not supported. No information is provided regarding the status of the Crosslink negotiation. 01b Crosslink negotiation resolved as an Upstream Port. 10b Crosslink negotiation resolved as a Downstream Port. 11b Crosslink negotiation is not completed. <p>Once a value of 01b or 10b is returned in this field, that value must continue to be returned while the Link is Up.</p> | RO |
| 10 | <p>Flit Mode Status – When <u>Flit Mode Supported</u> is Set, this bit when Set indicates that the Link is or will be operating in Flit Mode.</p> <p>For Downstream Ports, this bit is only meaningful when <u>Downstream Component Presence</u> is either 011b, 100b, or 101b. In all other states, <u>this bit must contain Zero</u>.</p> <p>For Upstream Ports, this bit is meaningful when the Link is Up. When the Link is Down, the value is implementation specific.</p> <p>This bit is RsvdZ if <u>Flit Mode Supported</u> is Clear.</p> | RO / RsvdZ |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|-------------------------------------|---|------------------|
| ↑↓11↑ ECN: Base 6.3 Optical△◀ | <p>↑↓Optical Retimer Presence Detected – When set to 1b, this bit indicates that an Optical Retimer Solution was present during the most recent Link negotiation. Refer to § Section 4.2.7.3.3.1 and § Section 4.2.7.3.3.2 for details.↑</p> <p>↑↓The default value of this bit is 0b.↑</p> <p>↑↓This bit is required for Ports that have the Optical Retimer Presence Detect Supported bit of the Link Capabilities 2 Register set to 1b.↑</p> <p>↑↓Ports that have the Optical Retimer Presence Detect Supported bit set to 0b are permitted to hardwire this bit to 0b.↑</p> <p>↑↓For Multi-Function Devices associated with an Upstream Port, this bit must be implemented in Function 0 and RsvdZ in all other functions.↑</p> | ↑↓ROS/ RsvdZ↑ |
| 14:12 | <p>Downstream Component Presence - This field indicates the presence and DRS status for the Downstream Component, if any, connected to the Link; defined values are:</p> <ul style="list-style-type: none"> 000b Link Down - Presence Not Determined 001b Link Down - Component Not Present indicates the Downstream Port (DSP) has determined that a Downstream Component is not present 010b Link Down - Component Present indicates the DSP has determined that a Downstream Component is present, but the Data Link Layer is not active 011b Link Down - Flit Mode Negotiation Completed indicates that the DSP's LTSSM has determined whether or not the Link will be operating in Flit Mode, but the Data Link Layer is not yet active. The Flit Mode Status bit is meaningful in this state. 100b Link Up - Component Present indicates the DSP has determined that a Downstream Component is present, but no DRS Message has been received since the Data Link Layer became active. The Flit Mode Status bit is meaningful in this state. 101b Link Up - Component Present and DRS Received indicates the DSP has received a DRS Message since the Data Link Layer became active. The Flit Mode Status bit is meaningful in this state. 110b Reserved 111b Reserved <p>The “Component Present” portion of the Downstream Component Presence field state must be determined by the logical “OR” of the Physical Layer in-band presence detect (in-band PD) mechanism and any out-of-band presence detect (OOB PD) mechanism supported by the Link. I.e., the Component Present state is true if either mechanism indicates a component is present. If no OOB PD mechanism is supported, then the Component Present state must be determined solely by the in-band PD mechanism. If the In-Band PD Disable bit in the Slot Control Register is Set, the in-band PD mechanism must always indicate that no component is present.</p> <p>For this field, the Component Present state is not always logically consistent with the Link Up state. As covered in the following paragraph, race conditions can result in Link Up being true while Component Present is false. This temporary condition¹⁸² cannot be accurately indicated using the field's architected encodings. It is strongly recommended that this condition be indicated using the encoding 001b (Link Down - Component Not Present).</p> <p>When a slot supports OOB PD, in-band PD is disabled, and an async removal or async hot-add is performed¹⁸³, race conditions may result in the DSP's LTSSM indicating that the Link is up while the OOB PD mechanism indicates that no component is present. For an async removal, this is usually caused by the LTSSM not detecting Link</p> | RO / RsvdZ |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|-----------------|
| | <p>Down immediately. For an async hot-add, this is usually caused by the OOB PD mechanism not immediately indicating that the component is present. For either case, it's less likely to cause software issues by the field value being 001b during this condition.</p> <p>Component Presence, Link Up, and DRS Received states indicated by this field must reflect their maskable states, which are controlled by the SFI PD State Mask , SFI DLL State Mask , or SFI DRS Mask bits in the SFI Control Register . See § Section 7.9.22.3 .</p> <p>This field must be implemented in any Downstream Port where the DRS Supported bit is Set in the Link Capabilities 2 Register . This field must be implemented in any Downstream Port where the Flit Mode Supported bit is Set.</p> <p>This field is RsvdZ for all other Functions.</p> <p>Default value of this field is 000b.</p> | |
| 15 | <p>DRS Message Received - This bit must be Set whenever the Port receives a DRS Message.</p> <p>This bit must be Cleared in DL_Down .</p> <p>This bit must be implemented in any Downstream Port where the DRS Supported bit is Set in the Link Capabilities 2 Register .</p> <p>This bit is RsvdZ for all other Functions.</p> <p>Default value of this bit is 0b.</p> | RW1C / RsvdZ |

7.5.3.21 Slot Capabilities 2 Register (Offset 34h) §

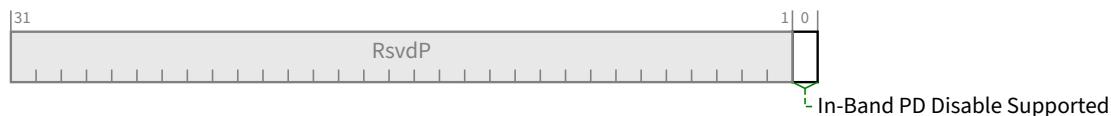


Figure 7-41 Slot Capabilities 2 Register §

Table 7-38 Slot Capabilities 2 Register §

| Bit Location | Register Description | Attributes |
|------------------------------|--|------------|
| 0 | <p>In-Band PD Disable Supported When Set, this bit indicates that this slot supports disabling the reporting of the in-band presence detect state, as controlled by the In-Band PD Disable bit in the Slot Control Register . If the slot does not support an out-of-band presence detect mechanism, this bit must be Clear.</p> | HwInit |
| ↑↑2:1↑ ECN: Base 6.3 eSFI△◀▶ | <p>↑↑SCap2 OOB PD Supported – This field provides similar functionality to the SFI OOB PD Supported bit and is intended for use when the SFI Extended Capability is either not implemented or is hidden when accessed by Configuration Read Requests or Configuration Write Requests (e.g., initiated by in-band management system software). See the SFI Hidden In-Band bit.↑</p> | ↑↑HwInit↑ |

182. This is a temporary condition and not an indefinite one. Notably, if the slot does not support OOB PD, and software attempts to Set the In-Band PD Disable bit, the bit will remain Clear since the In-Band PD Disable Supported bit must be Clear and the In-Band PD Disable bit must be hardwired to 0b

183. See IMPLEMENTATION NOTE: IN-BAND PRESENCE DETECT MECHANISM DEPRECATED FOR ASYNC HOT-PLUG .

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b This field does not indicate if OOB PD is supported or not. 01b Reserved 10b OOB PD is not supported for this slot. 11b OOB PD is supported for this slot. <p>If this Downstream Port has no implemented slot (as indicated by the Slot Implemented bit), then the value of this field must be 00b or 10b.</p> | |

7.5.3.22 Slot Control 2 Register (Offset 38h) §

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdP .

7.5.3.23 Slot Status 2 Register (Offset 3Ah) §

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdZ .

7.6 PCI Express Extended Capabilities §

PCI Express Extended Capability registers are located in Configuration Space at offsets 256 or greater as shown in § Figure 7-42 or in the Root Complex Register Block (RCRB). These registers when located in the Configuration Space are accessible using only the PCI Express Enhanced Configuration Access Mechanism (ECAM).

PCI Express Extended Capability structures are allocated using a linked list of optional or required PCI Express Extended Capabilities following a format resembling PCI Capability structures. The first DWORD of the Capability structure identifies the Capability and version and points to the next Capability as shown in § Figure 7-42 .

Each Capability structure must be DWORD aligned.

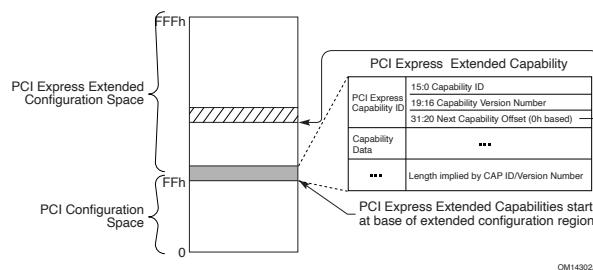


Figure 7-42 PCI Express Extended Configuration Space Layout §

7.6.1 Extended Capabilities in Configuration Space §

Extended Capabilities in Configuration Space always begin at offset 100h with a PCI Express Extended Capability header (§ Section 7.6.3). Absence of any Extended Capabilities is required to be indicated by an Extended Capability header with a Capability ID of 0000h, a Capability Version of 0h, and a Next Capability Offset of 000h.

7.6.2 Extended Capabilities in the Root Complex Register Block §

Extended Capabilities in a Root Complex Register Block always begin at offset 000h with a PCI Express Extended Capability header (§ Section 7.6.3). Absence of any Extended Capabilities is required to be indicated by an Extended Capability header with a Capability ID of FFFFh and a Next Capability Offset of 000h.

7.6.3 PCI Express Extended Capability Header §

All PCI Express Extended Capabilities must begin with a PCI Express Extended Capability Header. § Figure 7-43 details the allocation of register fields of a PCI Express Extended Capability Header; § Table 7-39 provides the respective bit definitions.

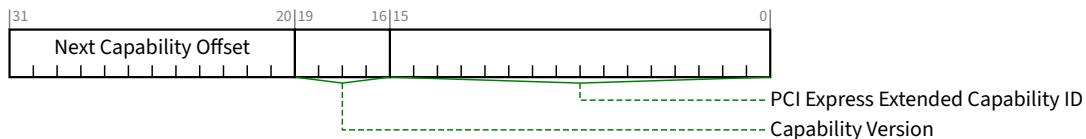


Figure 7-43 PCI Express Extended Capability Header §

Table 7-39 PCI Express Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. A version of the specification that changes the Extended Capability in a way that is not otherwise identifiable (e.g., through a new Capability field) is permitted to increment this field. All such changes to the Capability structure must be software-compatible. Software must check for Capability Version numbers that are greater than or equal to the highest number defined when the software is written, as Functions reporting any such Capability Version numbers will contain a Capability structure that is compatible with that piece of software. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits. | |

7.7 PCI and PCIe Capabilities Required by the Base Spec in Some Situations §

The following capabilities are required by this specification for some Functions. For example, Functions that support specific data rates, functions that generate interrupts, etc.

7.7.1 MSI Capability Structures §

All PCI Express Endpoint Functions that are capable of generating interrupts must implement MSI or MSI-X or both.

The MSI Capability structure is described in this section. The MSI-X Capability structure is described in § Section 7.7.2 .

The MSI Capability structure is illustrated in § Figure 7-44 and § Figure 7-45 . Each device Function that supports MSI (in a Multi-Function Device) must implement its own MSI Capability structure. More than one MSI Capability structure per Function is prohibited, but a Function is permitted to have both an MSI and an MSI-X Capability structure.

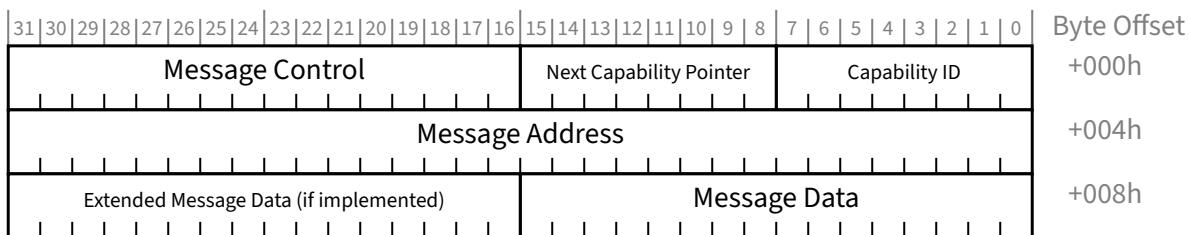


Figure 7-44 MSI Capability Structure for 32-bit Message Address §

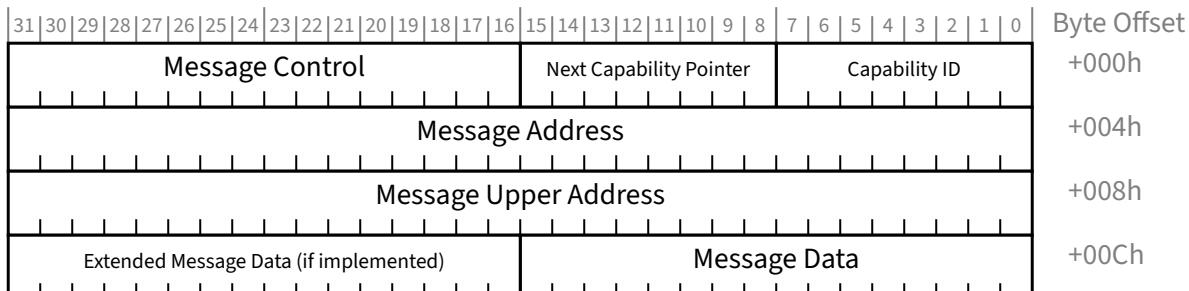


Figure 7-45 MSI Capability Structure for 64-bit Message Address §

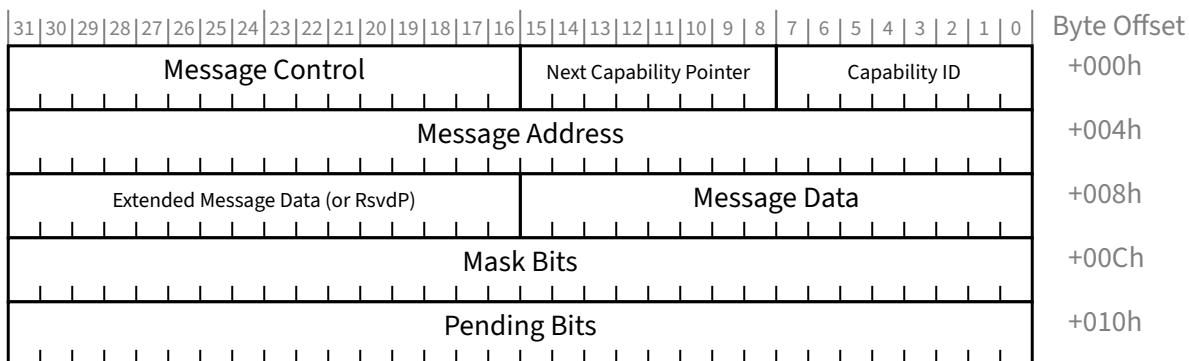


Figure 7-46 MSI Capability Structure for 32-bit Message Address and PVM §

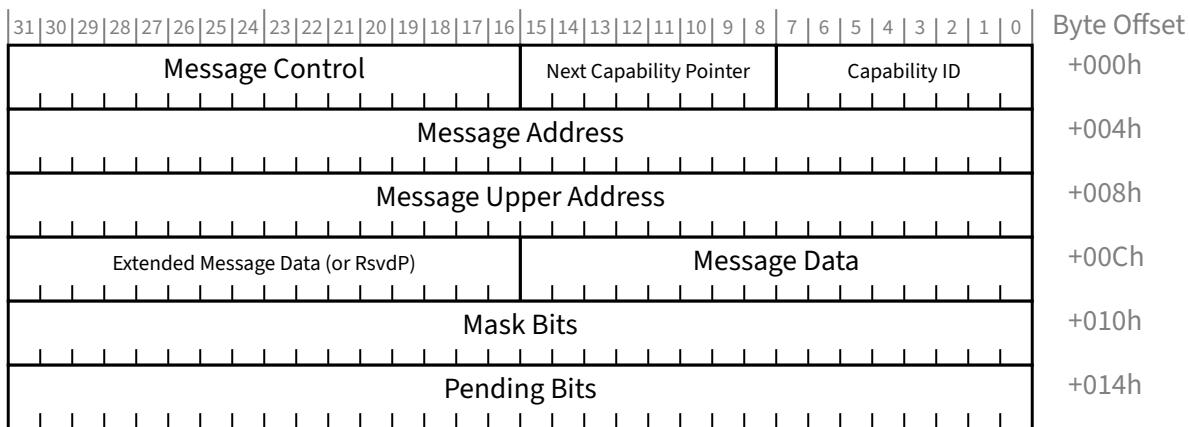


Figure 7-47 MSI Capability Structure for 64-bit Message Address and PVM §

To request service, an MSI Function writes the contents of the Message Data Register for MSI, and if enabled, the Extended Message Data Register for MSI, to the address specified by the contents of the Message Address Register for MSI (and, optionally, when 64-bit message addresses are used, the Message Upper Address Register for MSI). A read of the address specified by the contents of the Message Address register produces undefined results.

A Function supporting MSI implements one of four MSI Capability structure layouts illustrated in § Figure 7-44 to § Figure 7-47, depending upon which optional features are supported. A Legacy Endpoint that implements MSI is required to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure. A PCI Express Endpoint that implements MSI is required to support the 64-bit Message Address version of the MSI Capability structure. The Message Control Register for MSI indicates the Function's capabilities and provides system software control over MSI.

Each field is further described in the following sections.

7.7.1.1 MSI Capability Header (Offset 00h) §

The MSI Capability Header enumerates the MSI Capability structure in the PCI Configuration Space Capability list. § Figure 7-48 details allocation of register fields in the MSI Capability Header; § Table 7-40 provides the respective bit definitions.

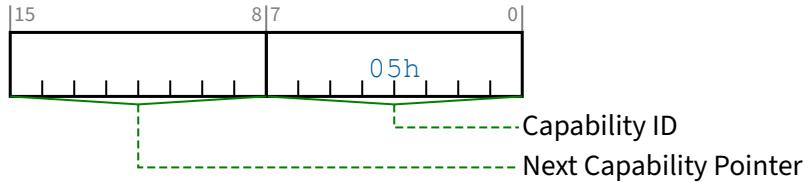


Figure 7-48 MSI Capability Header §

Table 7-40 MSI Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | Capability ID - Indicates the MSI Capability structure. This field must return a Capability ID of 05h indicating that this is an MSI Capability structure. | RO |
| 15:8 | Next Capability Pointer - This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities. | RO |

7.7.1.2 Message Control Register for MSI (Offset 02h) §

This register provides system software control over MSI. By default, MSI is disabled. If MSI and MSI-X are both disabled, the Function requests servicing using INTx interrupts (if supported). System software can enable MSI by Setting bit 0 of this register. System software is permitted to modify the Message Control Register for MSI's read-write bits and fields. A device driver is not permitted to modify the Message Control Register for MSI's read-write bits and fields.

Base 6.4 vs Base 6.3

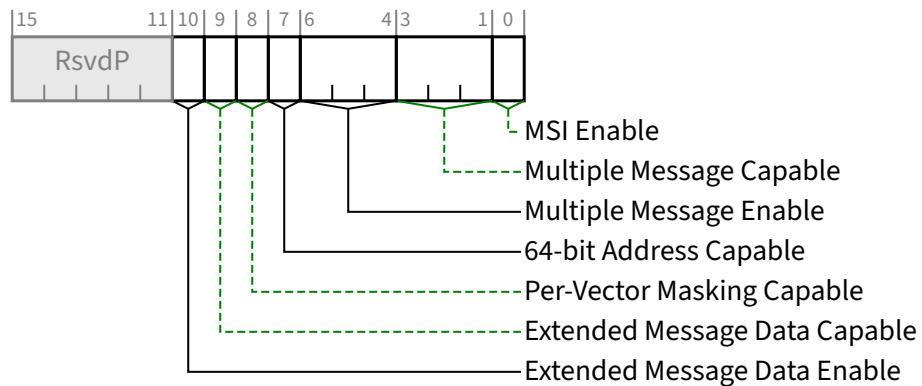


Figure 7-49 Message Control Register for MSI §

Table 7-41 Message Control Register for MSI §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|-------------|--------------------|-------------|---------------------|-------------|---------------------|-------------|---------------------|-------------|----------------------|-------------|----------------------|-------------|----------|-------------|----------|----|
| 0 | <p>MSI Enable - If Set and the MSI-X Enable bit in the Message Control Register for MSI-X (see § Section 7.7.2.2) is Clear, the Function is permitted to use MSI to request service and is prohibited from using INTx interrupts. System configuration software Sets this bit to enable MSI. Refer to § Section 7.5.1.1.3 for control of INTx interrupts.</p> <p>If Clear, the Function is prohibited from using MSI to request service.</p> <p>Software changing this bit during active operation may result in the Function dropping pending interrupt conditions or failing to recognize new interrupt conditions. See § Section 6.1.4.5 .</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | | | | | | | | | |
| 3:1 | <p>Multiple Message Capable - System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two (if a Function requires three vectors, it requests four by initializing this field to 010b). The encoding is defined as:</p> <table> <tbody> <tr> <td>000b</td><td>1 vector requested</td></tr> <tr> <td>001b</td><td>2 vectors requested</td></tr> <tr> <td>010b</td><td>4 vectors requested</td></tr> <tr> <td>011b</td><td>8 vectors requested</td></tr> <tr> <td>100b</td><td>16 vectors requested</td></tr> <tr> <td>101b</td><td>32 vectors requested</td></tr> <tr> <td>110b</td><td>Reserved</td></tr> <tr> <td>111b</td><td>Reserved</td></tr> </tbody> </table> | 000b | 1 vector requested | 001b | 2 vectors requested | 010b | 4 vectors requested | 011b | 8 vectors requested | 100b | 16 vectors requested | 101b | 32 vectors requested | 110b | Reserved | 111b | Reserved | RO |
| 000b | 1 vector requested | | | | | | | | | | | | | | | | | |
| 001b | 2 vectors requested | | | | | | | | | | | | | | | | | |
| 010b | 4 vectors requested | | | | | | | | | | | | | | | | | |
| 011b | 8 vectors requested | | | | | | | | | | | | | | | | | |
| 100b | 16 vectors requested | | | | | | | | | | | | | | | | | |
| 101b | 32 vectors requested | | | | | | | | | | | | | | | | | |
| 110b | Reserved | | | | | | | | | | | | | | | | | |
| 111b | Reserved | | | | | | | | | | | | | | | | | |
| 6:4 | <p>Multiple Message Enable - software writes to this field to indicate the number of allocated vectors. The number of allocated vectors is aligned to a power of two. As an example, if a Function requests four vectors (indicated by a Multiple Message Capable encoding of 010b), software can allocate either four, two, or one vector by writing a 010b, 001b, or 000b to this field, respectively.</p> <p>Behavior is undefined if the number of vectors allocated is greater than the number of vectors requested.</p> <p>Behavior is undefined if this field is changed while <u>MSI Enable</u> is Set.</p> | RW | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>When MSI Enable is Set, a Function will be allocated at least 1 vector. The encoding is defined as:</p> <ul style="list-style-type: none"> 000b 1 vector allocated 001b 2 vectors allocated 010b 4 vectors allocated 011b 8 vectors allocated 100b 16 vectors allocated 101b 32 vectors allocated 110b Reserved 111b Reserved <p>Function behavior is undefined if software changes the value of this field while the MSI Enable bit is Set. Default value of this field is 000b.</p> | |
| 7 | 64-bit Address Capable - If Set, the Function is capable of sending a 64-bit Message Address. If Clear, the Function is not capable of sending a 64-bit Message Address. This bit must be Set if the Function is a PCI Express Endpoint, as indicated by the value in the Device/Port Type field. This bit <i>MUST</i> be Set. | RO |
| 8 | Per-Vector Masking Capable - If Set, the Function supports MSI Per-Vector Masking. If Clear, the Function does not support MSI Per-Vector Masking. This bit must be Set if the Function is a PF or VF within an SR-IOV Device. This bit is permitted to be Set or Clear if the Function is an SIOV PF. | RO |
| 9 | Extended Message Data Capable - If Set, the Function is capable of providing Extended Message Data . If Clear, the Function does not support providing Extended Message Data . | RO |
| 10 | Extended Message Data Enable - If Set, the Function is enabled to provide Extended Message Data . If Clear, the Function is not enabled to provide Extended Message Data . Default value of this bit is 0b. This bit must be read-write if the Extended Message Data Capable bit is 1b; otherwise it must be hardwired to 0b. | RW / RO |

7.7.1.3 Message Address Register for MSI (Offset 04h) §

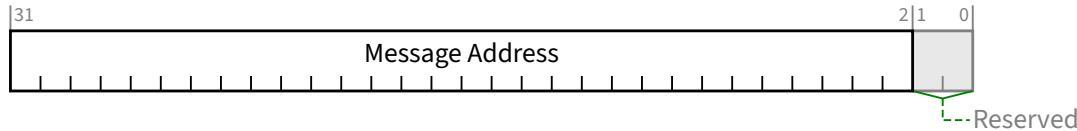


Figure 7-50 Message Address Register for MSI §

Table 7-42 Message Address Register for MSI §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1:0 | Reserved Reserved - Always returns 0 on read. Write operations have no effect. | RsvdP |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:2 | <p>Message Address - System-specified message address.</p> <p>If the MSI Enable bit is Set, the contents of this register specify the DWORD-aligned address (Address[31:02]) for the MSI transaction. Address[1:0] are set to 00b.</p> <p>Default value of this field is undefined.</p> | RW |

7.7.1.4 Message Upper Address Register for MSI (Offset 08h) §

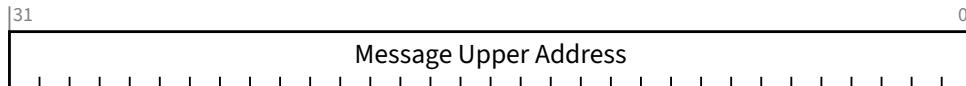


Figure 7-51 Message Upper Address Register for MSI §

Table 7-43 Message Upper Address Register for MSI §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:0 | <p>Message Upper Address - System-specified message upper address.</p> <p>This register is implemented only if the Function supports a 64-bit message address (64-bit Address Capable is Set). This register is implemented only if the Function supports a 64-bit message address (64-bit Address Capable is Set). This register is required for PCI Express Endpoints (as indicated by the value in the Device/Port Type field) and is optional for other Function types.</p> <p>If the MSI Enable bit is Set, the contents of this register (if non-zero) specify the upper 32-bits of a 64-bit message address (Address[63:32]). If the contents of this register are zero, the Function uses the 32 bit address specified by the Message Address register.</p> <p>Default value of this field is undefined.</p> | RW |

7.7.1.5 Message Data Register for MSI (Offset 08h or 0Ch) §

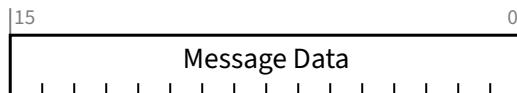


Figure 7-52 Message Data Register for MSI §

Table 7-44 Message Data Register for MSI §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | <p>Message Data - System-specified message data.</p> <p>If the MSI Enable bit is Set, the Function sends a DWORD Memory Write transaction using Message Data for the lower 16 bits. All 4 Byte Enables are Set.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>The Multiple Message Enable field defines the number of low order message data bits the Function is permitted to modify to generate its system software allocated vectors. For example, a Multiple Message Enable encoding of 010b indicates the Function has been allocated four vectors and is permitted to modify message data bits 1 and 0 (a Function modifies the lower message data bits to generate the allocated number of vectors). If the Multiple Message Enable field is 000b, the Function is not permitted to modify the message data. When Multiple Message Enable is non-zero, behavior is undefined if the corresponding low order bits of this register are not 0b.</p> <p>Default value of this field is undefined.</p> | |

7.7.1.6 Extended Message Data Register for MSI (Optional) §

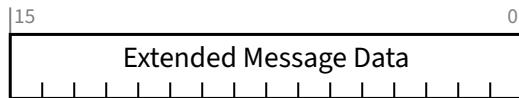


Figure 7-53 Extended Message Data Register for MSI §

Table 7-45 Extended Message Data Register for MSI §

| Bit Location | Register Description | Attributes |
|--------------|---|----------------------------|
| 15:0 | <p>Extended Message Data - System-specified message data.</p> <p>This register is optional. For the MSI Capability structures without Per-vector Masking, it must be implemented if the Extended Message Data Capable bit is Set; otherwise, it is outside the MSI Capability structure and undefined. For the MSI Capability structures with Per-vector Masking, it must be implemented if the Extended Message Data Capable bit is Set; otherwise, it is RsvdP.</p> <p>If the Extended Message Data Enable bit is Set, the DWORD Memory Write transaction uses Extended Message Data for the upper 16 bits; otherwise, it uses 0000h for the upper 16 bits.</p> <p>Default value of this field is 0000h.</p> | RW /undefined/ RsvdP |

7.7.1.7 Mask Bits Register for MSI (Offset 0Ch or 10h) §

This register is optional. It is present if Per-Vector Masking Capable is Set (see § Section 7.7.1.2). The offset of this register within the capability depends on the value of the 64-bit Address Capable bit (see § Section 7.7.1.2).

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis.

MSI vectors are numbered 0 through N-1, where N is the number of vectors allocated by software. Each vector is associated with a correspondingly numbered bit in the Mask Bits and Pending Bits registers.

The Multiple Message Capable field indicates how many vectors (with associated Mask and Pending bits) are implemented. All unimplemented Mask and Pending bits are Reserved.

The Multiple Message Enable field controls how many vectors are allocated for use. The value of each implemented Mask bit and Pending bit that is currently not allocated must be ignored by hardware; i.e., the value must not affect the generation of interrupts.

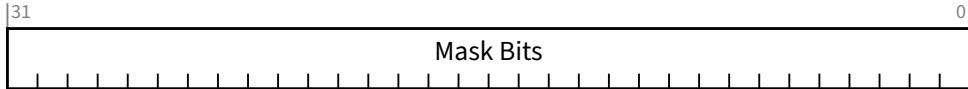


Figure 7-54 Mask Bits Register for MSI §

Table 7-46 Mask Bits Register for MSI §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:0 | Mask Bits - For each Mask bit that is Set, the Function is prohibited from sending the associated message. Default is 0. | RW |

7.7.1.8 Pending Bits Register for MSI (Offset 10h or 14h) §

This register is optional. It is present if Per-Vector Masking Capable is Set (see § Section 7.7.1.2).

The offset of this register within the capability depends on the value of the 64-bit Address Capable bit (see § Section 7.7.1.2)

See § Section 7.7.1.7 for additional requirements on this register.

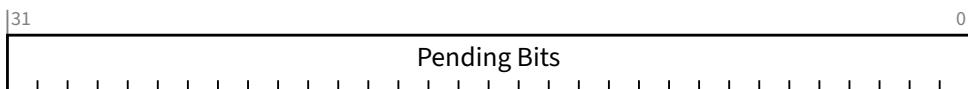


Figure 7-55 Pending Bits Register for MSI §

Table 7-47 Pending Bits Register for MSI §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:0 | Pending Bits - For each Pending bit that is Set, the Function has a pending associated message. Default is 0. | RO |

7.7.2 MSI-X Capability and Table Structure §

The MSI-X Capability structure is illustrated in § Figure 7-56 . More than one MSI-X Capability structure per Function is prohibited, but a Function is permitted to have both an MSI Capability structure and an MSI-X Capability structure.

In contrast to the MSI Capability structure, which directly contains all of the control/status information for the Function's vectors, the MSI-X Capability structure instead points to an **MSI-X Table** structure and an **MSI-X PBA** structure (Pending Bit Array structure), each residing in Memory Space (see § Figure 7-57 and § Figure 7-58).

Each structure is mapped by a Base Address Register (BAR) belonging to the Function, located beginning at 10h in Configuration Space, or an entry in the Enhanced Allocation capability . A BAR Indicator register (BIR) indicates which BAR(or BEI when using Enhanced Allocation), and a QWORD-aligned Offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is permitted to be either 32-bit or 64-bit, but must map Memory Space. A Function is permitted to map both structures with the same BAR, or to map each structure with a different BAR.

The MSI-X Table structure, illustrated in § Figure 7-57 , typically contains multiple entries, each consisting of several fields: Message Address, Message Upper Address, Message Data, and Vector Control. Each entry is capable of specifying a unique vector.

The Pending Bit Array (PBA) structure, illustrated in § Figure 7-58 , contains the Function’s Pending Bits, one per Table entry, organized as a packed array of bits within QWORDS. The last QWORD will not necessarily be fully populated.

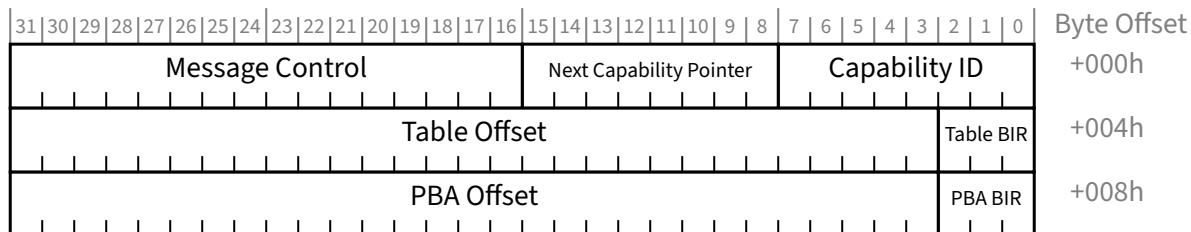


Figure 7-56 MSI-X Capability Structure §

Base 6.4 vs Base 6.3

| | Byte Offset |
|---|-------------|
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | +000h |
| Entry 0: Message Address | |
| Entry 0: Message Upper Address | +004h |
| Entry 0: Message Data | +008h |
| Entry 0: Vector Control | +00Ch |
| Entry 1: Message Address | +010h |
| Entry 1: Message Upper Address | +014h |
| Entry 1: Message Data | +018h |
| Entry 1: Vector Control | +01Ch |
| Entry 2: Message Address | +020h |
| Entry 2: Message Upper Address | +024h |
| Entry 2: Message Data | +028h |
| Entry 2: Vector Control | +02Ch |
| ... | +030h |

Figure 7-57 MSI-X Table Structure [§](#)

| 63 | 0 | | |
|--|----------------------|-------------------------|--|
| Pending Bits 0 through 63 | QWORD 0 | Base | |
| Pending Bits 64 through 127 | QWORD 1 | Base + 1*8 | |
| ... | ... | ... | |
| Pending Bits ((N-1) div 64)*64 through N-1 | QWORD ((N-1) div 64) | Base + ((N-1) div 64)*8 | |

A-0385

Figure 7-58 MSI-X PBA Structure [§](#)

To request service using a given MSI-X Table entry, a Function performs a DWORD Memory Write transaction using the contents of the Message Data field entry for data, the contents of the Message Upper Address field for the upper 32 bits of address, and the contents of the Message Address field entry for the lower 32 bits of address. A memory read transaction from the address targeted by the MSI-X message produces undefined results.

If a Base Address Register or entry in the Enhanced Allocation capability that maps address space for the MSI-X Table or MSI-X PBA also maps other usable address space that is not associated with MSI-X structures, locations (e.g., for CSRs) used in the other address space must not share any naturally aligned 4-KB address range with one where either MSI-X structure resides. This allows system software where applicable to use different processor attributes for MSI-X structures and the other address space. (Some processor architectures do not support having different processor attributes associated with the same naturally aligned 4-KB physical address range.) The MSI-X Table and MSI-X PBA are permitted to co-reside within a naturally aligned 4-KB address range, though they must not overlap with each other.

With ~~↑↓SR-IOV devices,↑↑SR-IOV Devices,↑~~ alignment requirements like those in the ~~↑↓preceding↓↑↑preceding↑~~ paragraph still apply, but they must be based on the System Page Size value from the PF's SR-IOV Extended Capability instead using a fixed 4-KB value.

IMPLEMENTATION NOTE: DEDICATED BARS AND ADDRESS RANGE ISOLATION §

To enable system software to map MSI-X structures onto different processor pages for improved access control, it is recommended that a Function dedicate separate Base Address Registers for the MSI-X Table and MSI-X PBA, or else provide more than the minimum required isolation with address ranges.

If dedicated separate Base Address Registers is not feasible, it is recommended that a Function dedicate a single Base Address Register for the MSI-X Table and MSI-X PBA.

If a dedicated Base Address Register is not feasible, it is recommended that a Function isolate the MSI-X structures from the non-MSI-X structures with aligned 8 KB ranges rather than the mandatory aligned 4 KB ranges.

For example, if a Base Address Register needs to map 2 KB for an MSI-X Table containing 128 entries, 16 bytes for an MSI-X PBA containing 128 bits, and 64 bytes for registers not related to MSI-X, the following is an acceptable implementation. The Base Address Register requests 8 KB of total address space, maps the first 64 bytes for the non MSI-X registers, maps the MSI-X Table beginning at an offset of 4 KB, and maps the MSI-X PBA beginning at an offset of 6 KB.

A preferable implementation for a shared Base Address Register is for it to request 16 KB of total address space, map the first 64 bytes for the non MSI-X registers, map the MSI-X Table beginning at an offset of 8 KB, and map the MSI-X PBA beginning at an offset of 12 KB.

IMPLEMENTATION NOTE: MSI-X MEMORY SPACE STRUCTURES IN READ/WRITE MEMORY §

The MSI-X Table and MSI-X PBA structures are defined such that they can reside in general purpose read/write memory on a device, for ease of implementation and added flexibility. To achieve this, none of the contained fields are required to be read-only, and there are also restrictions on transaction alignment and sizes.

For all accesses to MSI-X Table and MSI-X PBA fields, software must use aligned full DWORD or aligned full QWORD transactions; otherwise, the result is undefined.

MSI-X Table entries and Pending bits are each numbered 0 through N-1, where N-1 is indicated by the Table Size field in the Message Control Register for MSI-X. For a given arbitrary MSI-X Table entry k , its starting address can be calculated with the formula:

entry starting address = Table base + $k \times 16$

Equation 7-1 MSI-X Starting Address §

For the associated Pending bit k , its address for QWORD access and bit number within that QWORD can be calculated with the formulas:

QWORD address = PBA base + $(k \text{ div } 64) \times 8$

QWORD bit# = $k \text{ mod } 64$

Equation 7-2 MSI-X PBA QWORD Access §

Software that chooses to read Pending bit K with DWORD accesses can use these formulas:

DWORD address = PBA base + $(k \text{ div } 32) \times 4$

DWORD bit# = $k \text{ mod } 32$

Equation 7-3 MSI-X PBA DWORD Access §

Each field in the MSI-X Capability, MSI-X Table, and MSI-X PBA structures is further described in the following sections. Within the MSI-X Capability structure, Reserved registers and bits always return 0 when read, and write operations have no effect. Within the MSI-X Table and PBA structures, Reserved fields have special rules.

7.7.2.1 MSI-X Capability Header (Offset 00h) §

The MSI-X Capability Header enumerates the MSI-X Capability structure in the PCI Configuration Space Capability list. § Figure 7-56 details allocation of register fields in the MSI-X Capability Header; § Table 7-48 provides the respective bit definitions.

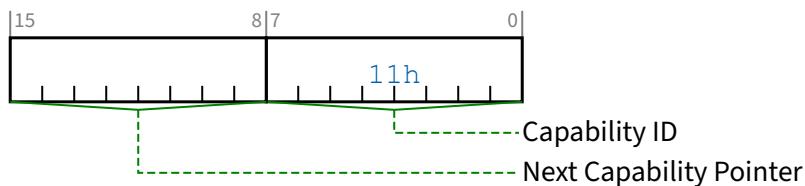


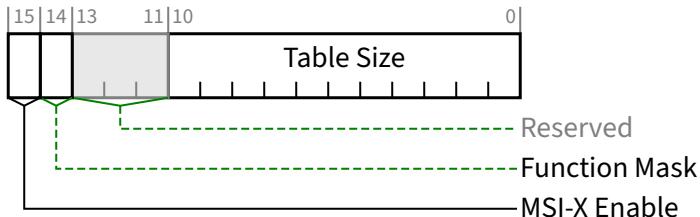
Figure 7-59 MSI-X Capability Header §

Table 7-48 MSI-X Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | Capability ID - Indicates the MSI-X Capability structure. This field must return a Capability ID of 11h indicating that this is an MSI-X Capability structure. | RO |
| 15:8 | Next Capability Pointer - This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities. | RO |

7.7.2.2 Message Control Register for MSI-X (Offset 02h) §

By default, MSI-X is disabled. If MSI and MSI-X are both disabled, the Function requests servicing via INTx interrupts (if supported). System software can enable MSI-X by Setting bit 15 of this register. System software is permitted to modify the Message Control register's read-write bits and fields. A device driver is not permitted to modify the Message Control register's read-write bits and fields.

*Figure 7-60 Message Control Register for MSI-X §**Table 7-49 Message Control Register for MSI-X §*

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 10:0 | Table Size - System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. For example, a returned value of 000 0000 0011b indicates a table size of 4. | RO |
| 13:11 | Reserved Reserved - Always returns 0 on a read, and a write operation has no effect. | RsvdP |
| 14 | Function Mask - If Set, all of the vectors associated with the Function are masked, regardless of their per-vector Mask bit values. If Clear, each vector's Mask bit determines whether the vector is masked or not. Setting or Clearing the MSI-X Function Mask bit has no effect on the value of the per-vector Mask bits. Default value of this bit is 0b. | RW |
| 15 | MSI-X Enable - If Set and the MSI Enable bit in the Message Control Register for MSI (see § Section 7.7.1.2) is Clear, the Function is permitted to use MSI-X to request service and is prohibited from using INTx interrupts (if implemented). System configuration software Sets this bit to enable MSI-X. If Clear, the Function is prohibited from using MSI-X to request service. Software changing this bit during active operation may result in the Function dropping pending interrupt conditions or failing to recognize new interrupt conditions. See § Section 6.1.4.5 . Default value of this bit is 0b. | RW |

7.7.2.3 Table Offset/Table BIR Register for MSI-X (Offset 04h) §

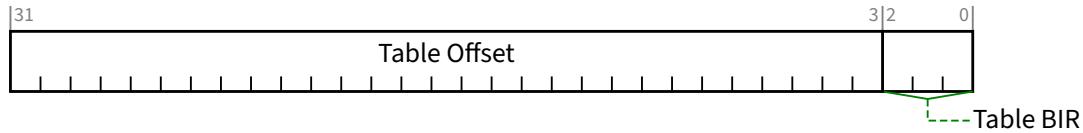


Figure 7-61 Table Offset/Table BIR Register for MSI-X §

Table 7-50 Table Offset/Table BIR Register for MSI-X §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>Table BIR - Indicates which one of a Function's Base Address Registers , located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BAR Equivalent Indicator (BEI) , is used to map the Function's MSI-X Table into Memory Space.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0 Base Address Register 10h 1 Base Address Register 14h 2 Base Address Register 18h 3 Base Address Register 1Ch 4 Base Address Register 20h 5 Base Address Register 24h 6 Reserved 7 Reserved <p>For a 64-bit Base Address Register , the Table BIR indicates the lower DWORD. For Functions with Type 1 Configuration Space headers, BIR values 2 through 5 are also Reserved.</p> | RO |
| 31:3 | <p>Table Offset - Used as an offset from the address contained by one of the Function's Base Address Registers to point to the base of the MSI-X Table . The lower 3 Table BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset.</p> <p>For VFs, the Table Offset value is relative to the VF's Memory address space.</p> | RO |

7.7.2.4 PBA Offset/PBA BIR Register for MSI-X (Offset 08h) §

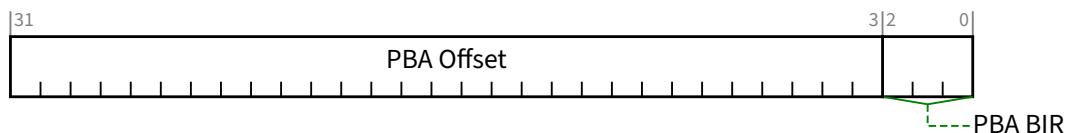


Figure 7-62 PBA Offset/PBA BIR Register for MSI-X §

Table 7-51 PBA Offset/PBA BIR Register for MSI-X §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>PBA BIR - Indicates which one of a Function's Base Address Registers , located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BEI, is used to map the Function's MSI-X PBA into Memory Space.</p> <p>The PBA BIR value definitions are identical to those for the Table BIR .</p> | RO |
| 31:3 | <p>PBA Offset - Used as an offset from the address contained by one of the Function's Base Address Registers to point to the base of the MSI-X PBA . The lower 3 PBA BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset.</p> <p>For VFs, the PBA Offset value is relative to the VF's Memory address space.</p> | RO |

7.7.2.5 Message Address Register for MSI-X Table Entries §

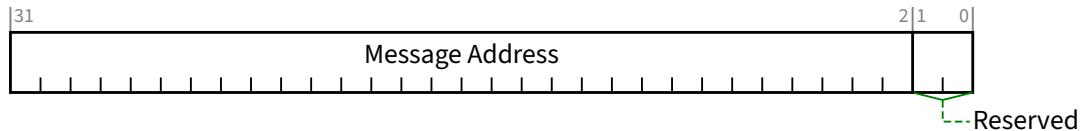


Figure 7-63 Message Address Register for MSI-X Table Entries §

Table 7-52 Message Address Register for MSI-X Table Entries §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1:0 | <p>Reserved</p> <p>Reserved - For proper DWORD alignment, software must always write zeros to these two bits; otherwise the result is undefined.</p> <p>Default value of this field is 00b.</p> <p>These bits are permitted to be read-only or read-write.</p> | RO or RW |
| 31:2 | <p>Message Address - System-specified message lower address.</p> <p>For MSI-X messages, the contents of this field from an MSI-X Table entry specifies the lower portion of the DWORD-aligned address for the Memory Write transaction.</p> <p>Default value of this field is undefined.</p> | RW |

7.7.2.6 Message Upper Address Register for MSI-X Table Entries §

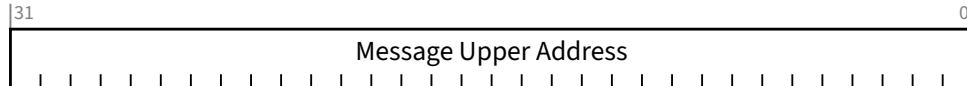


Figure 7-64 Message Upper Address Register for MSI-X Table Entries §

Table 7-53 Message Upper Address Register for MSI-X Table Entries §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:0 | <p>Message Upper Address - System-specified message upper address bits.</p> <p>If this field is zero, 32-bit address messages are used. If this field is non-zero, 64-bit address messages are used.</p> <p>Default value of this field is undefined.</p> | RW |

7.7.2.7 Message Data Register for MSI-X Table Entries §

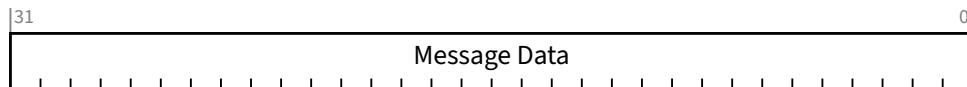


Figure 7-65 Message Data Register for MSI-X Table Entries §

Table 7-54 Message Data Register for MSI-X Table Entries §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:0 | <p>Message Data - System-specified message data.</p> <p>For MSI-X messages, the contents of this field from an MSI-X Table entry specifies the 32-bit data payload of the DWORD Memory Write transaction. All 4 Byte Enables are Set.</p> <p>In contrast to message data used for MSI messages, the low-order message data bits in MSI-X messages are not modified by the Function.</p> <p>This field is read-write.</p> <p>Default value of this field is undefined.</p> | RW |

7.7.2.8 Vector Control Register for MSI-X Table Entries §

If a Function implements a TPH Requester Extended Capability structure and an MSI-X Capability structure, the Function can optionally use the Vector Control Register for MSI-X Table Entries in each entry to store a Steering Tag. See § Section 6.17 .

Base 6.4 vs Base 6.3

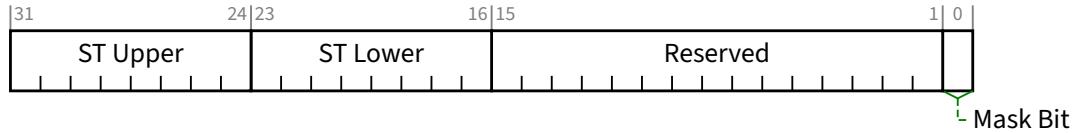


Figure 7-66 Vector Control Register for MSI-X Table Entries §

Table 7-55 Vector Control Register for MSI-X Table Entries §

| Bit Location | Register Description | Attributes |
|--------------|--|-------------|
| 0 | <p>Mask Bit - When this bit is Set, the Function is prohibited from sending a message using this MSI-X Table entry. However, any other MSI-X Table entries programmed with the same vector will still be capable of sending an equivalent message unless they are also masked.</p> <p>Default value of this bit is 1b (entry is masked)</p> | RW |
| 15:1 | <p>Reserved</p> <p>Reserved - By default, the value of these bits must be 0. However, for potential future use, software must preserve the value of these Reserved bits when modifying the value of other Vector Control bits. If software modifies the value of these Reserved bits, the result is undefined.</p> <p>These bits are permitted to be RsvdP or read-write.</p> | RW or RsvdP |
| 23:16 | <p>ST Lower - If the Function implements a TPH Requester Extended Capability structure, and the ST Table Location indicates a value of 10b, then this field contains the lower 8 bits of a Steering Tag and must be read-write.</p> <p>Otherwise, this field is permitted to be read-write or RsvdP , and for potential future use, software must preserve the value of these Reserved bits when modifying the value of other Vector Control bits, or the result is undefined.</p> <p>Default value of this field is 00h.</p> | RW / RsvdP |
| 31:24 | <p>ST Upper - If the Function implements a TPH Requester Extended Capability structure, and the ST Table Location indicates a value of 10b, and the <u>Extended TPH Requester Supported</u> bit is Set, then this field contains the upper 8 bits of a Steering Tag and must be read-write.</p> <p>Otherwise, this field is permitted to be read-write or RsvdP , and for potential future use, software must preserve the value of these Reserved bits when modifying the value of other Vector Control bits, or the result is undefined.</p> <p>Default value of this field is 00h.</p> | RW / RsvdP |

7.7.2.9 Pending Bits Register for MSI-X PBA Entries §

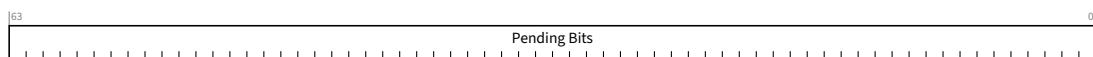


Figure 7-67 Pending Bits Register for MSI-X PBA Entries §

Table 7-56 Pending Bits Register for MSI-X PBA Entries §

| Bit Location | Register Description | Attributes |
|--------------|--|------------------------|
| 63:0 | <p>Pending Bits - For each Pending Bit that is Set, the Function has a pending message for the associated MSI-X Table entry.</p> <p>Pending bits that have no associated <u>MSI-X Table</u> entry are Reserved. By default, the value of Reserved Pending bits must be 0b.</p> <p>Software should never write, and should only read Pending Bits. If software writes to Pending Bits, the result is undefined.</p> <p>Default value of each Pending Bit is 0b.</p> <p>These bits are permitted to be read-only or read-write.</p> | <u>RO</u> or <u>RW</u> |

7.7.3 Secondary PCI Express Extended Capability §

The Secondary PCI Express Extended Capability structure must be implemented in any Function or RCRB where any of the following are true:

- The Supported Link Speeds Vector field indicates that the Link supports Link Speeds of 8.0 GT/s or higher (see § Section 7.5.3.18 or § Section 7.9.9.2).
- Any bit in the Lower SKP OS Generation Supported Speeds Vector field is Set (see § Section 7.5.3.18).
- When Lane based errors are reported in the Lane Error Status register (discussed in § Section 4.2.7).

To support future additions to this capability, this capability is permitted in any Function or RCRB associated with a Link. For a Multi-Function Device associated with an Upstream Port, this capability is permitted only in Function 0 of the Device.

Base 6.4 vs Base 6.3

| Byte Offset | |
|-------------|---|
| +000h | PCI Express Extended Capability Header |
| +004h | Link Control 3 Register |
| +008h | Lane Error Status Register |
| +00Ch | Lane (1) Equalization Control Register Entry |
| +010h | Lane (0) Equalization Control Register Entry |
| +014h | Lane (3) Equalization Control Register Entry |
| +018h | Lane (2) Equalization Control Register Entry |
| +01Ch | Lane (5) Equalization Control Register Entry |
| +01Eh | Lane (4) Equalization Control Register Entry |
| +020h | Lane (7) Equalization Control Register Entry |
| +024h | Lane (6) Equalization Control Register Entry |
| +028h | Lane (9) Equalization Control Register Entry |
| +02Ch | Lane (8) Equalization Control Register Entry |
| +030h | Lane (11) Equalization Control Register Entry |
| +034h | Lane (10) Equalization Control Register Entry |
| +038h | Lane (13) Equalization Control Register Entry |
| +03Ch | Lane (12) Equalization Control Register Entry |
| +040h | Lane (15) Equalization Control Register Entry |
| +044h | Lane (14) Equalization Control Register Entry |
| +048h | Lane (17) Equalization Control Register Entry |
| | Lane (16) Equalization Control Register Entry |
| | Lane (19) Equalization Control Register Entry |
| | Lane (18) Equalization Control Register Entry |
| | Lane (21) Equalization Control Register Entry |
| | Lane (20) Equalization Control Register Entry |
| | Lane (23) Equalization Control Register Entry |
| | Lane (22) Equalization Control Register Entry |
| | Lane (25) Equalization Control Register Entry |
| | Lane (24) Equalization Control Register Entry |
| | Lane (27) Equalization Control Register Entry |
| | Lane (26) Equalization Control Register Entry |
| | Lane (29) Equalization Control Register Entry |
| | Lane (28) Equalization Control Register Entry |
| | Lane (31) Equalization Control Register Entry |
| | Lane (30) Equalization Control Register Entry |

Figure 7-68 Secondary PCI Express Extended Capability Structure §

7.7.3.1 Secondary PCI Express Extended Capability Header (Offset 00h) §

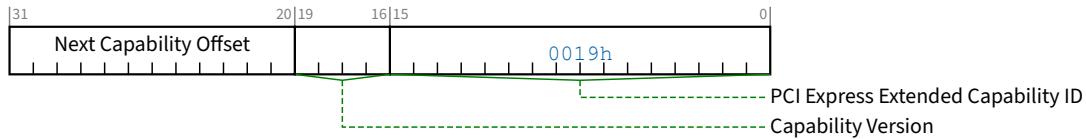


Figure 7-69 Secondary PCI Express Extended Capability Header §

Table 7-57 Secondary PCI Express Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Secondary PCI Express Extended Capability is 0019h. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

7.7.3.2 Link Control 3 Register (Offset 04h) §

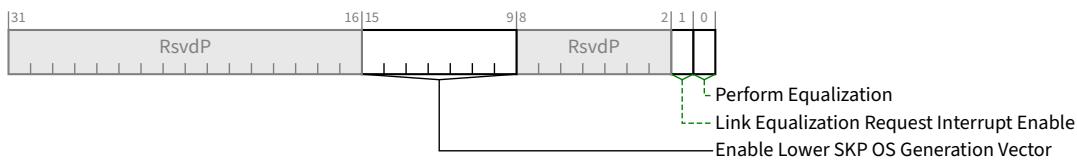


Figure 7-70 Link Control 3 Register §

Table 7-58 Link Control 3 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Perform Equalization - When this bit is 1b and a 1b is written to the Retrain Link bit with the Target Link Speed field set to 8.0 GT/s or higher, the Downstream Port must perform Link Equalization. Refer to § Section 4.2.4 and § Section 4.2.7.4.2 for details. This bit is RW for Downstream Ports and for Upstream Ports when Crosslink Supported is 1b (see § Section 7.5.3.18). This bit is not applicable and is RsvdP for Upstream Ports when the Crosslink Supported bit is 0b. The default value is 0b. If the Port does not support 8.0 GT/s, this bit is permitted to be hardwired to 0b. | RW / RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | |
|--------------|---|--------------|----------|--------------|----------|--------------|----------|--------------|-----------|--------------|-----------|--------------|-----------|--------------|-------|------------|
| 1 | <p>Link Equalization Request Interrupt Enable - When Set, this bit enables the generation of an interrupt to indicate that the Link Equalization Request 8.0 GT/s bit, the Link Equalization Request 16.0 GT/s bit, the Link Equalization Request 32.0 GT/s bit, or the Link Equalization Request 64.0 GT/s bit has been Set.</p> <p>The interrupt vector is Interrupt Message Number (see § Section 7.5.3.2).</p> <p>This bit is RW for Downstream Ports and for Upstream Ports when Crosslink Supported is 1b (see § Section 7.5.3.18). This bit is not applicable and is RsvdP for Upstream Ports when the Crosslink Supported bit is 0b.</p> <p>The default value for this bit is 0b.</p> <p>If the Port does not support 8.0 GT/s, this bit is permitted to be hardwired to 0b.</p> | RW / RsvdP | | | | | | | | | | | | | | |
| 9:15 | <p>Enable Lower SKP OS Generation Vector - When the Link is in L0 and the bit in this field corresponding to the Current Link Speed is Set, SKP Ordered Sets are scheduled at the rate defined for SRNS , overriding the rate required based on the clock tolerance architecture. See § Section 4.2.8 and § Section 4.2.8.4 for additional requirements.</p> <p>Bit definitions within this field are:</p> <table> <tr><td>Bit 0</td><td>2.5 GT/s</td></tr> <tr><td>Bit 1</td><td>5.0 GT/s</td></tr> <tr><td>Bit 2</td><td>8.0 GT/s</td></tr> <tr><td>Bit 3</td><td>16.0 GT/s</td></tr> <tr><td>Bit 4</td><td>32.0 GT/s</td></tr> <tr><td>Bit 5</td><td>64.0 GT/s</td></tr> <tr><td>Bit 6</td><td>RsvdP</td></tr> </table> <p>Each unreserved bit in this field must be RW if the corresponding bit in the Lower SKP OS Generation Supported Speeds Vector is Set, otherwise the bit must be RW or hardwired to 0.</p> <p>Behavior is undefined if a bit is Set in this field and the corresponding bit in the Lower SKP OS Generation Supported Speeds Vector is not Set.</p> <p>The default value of this field is 000 0000b.</p> | Bit 0 | 2.5 GT/s | Bit 1 | 5.0 GT/s | Bit 2 | 8.0 GT/s | Bit 3 | 16.0 GT/s | Bit 4 | 32.0 GT/s | Bit 5 | 64.0 GT/s | Bit 6 | RsvdP | RW / RsvdP |
| Bit 0 | 2.5 GT/s | | | | | | | | | | | | | | | |
| Bit 1 | 5.0 GT/s | | | | | | | | | | | | | | | |
| Bit 2 | 8.0 GT/s | | | | | | | | | | | | | | | |
| Bit 3 | 16.0 GT/s | | | | | | | | | | | | | | | |
| Bit 4 | 32.0 GT/s | | | | | | | | | | | | | | | |
| Bit 5 | 64.0 GT/s | | | | | | | | | | | | | | | |
| Bit 6 | RsvdP | | | | | | | | | | | | | | | |

7.7.3.3 Lane Error Status Register (Offset 08h) §

The Lane Error Status Register consists of a 32-bit vector, where each bit indicates if the Lane with the corresponding Lane number detected an error. This Lane number is the ~~default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.~~ ↑↑Physical Lane Number .↑

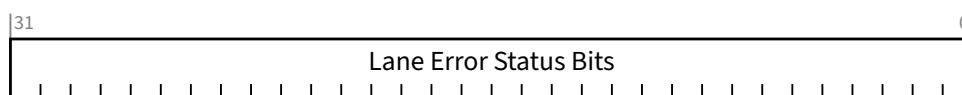
 Errata: Base 6.3
 B825△◀▷


Figure 7-71 Lane Error Status Register §

Table 7-59 Lane Error Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| 31:0 | <p>Lane Error Status Bits - Each bit indicates if the corresponding Lane detected a Lane-based error. A value of 1b indicates that a Lane based-error was detected on the corresponding Lane Number (see § Section 4.2.2.3.3 , § Section 4.2.2.3.4 , § Section 4.2.7 , and § Section 4.2.8.2 for details).</p> <p>The default value of each bit is 0b.</p> <p>For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are <u>RsvdZ</u>.</p> <p>For Ports that do not support 8.0 GT/s and do not set these bits based on 8b/10b errors (optional, see § Section 4.2.7), this field is permitted to be hardwired to 0.</p> | <u>RW1CS</u> |

7.7.3.4 Lane Equalization Control Register (Offset 0Ch) §

The Lane Equalization Control Register consists of control fields required for per-Lane 8.0 GT/s equalization and the number of entries in this register are sized by Maximum Link Width (see § Section 7.5.3.6). Each entry contains the values for the Lane with the corresponding ~~default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.~~ ~~Physical Lane Number~~

Errata: Base 6.3
B825△◀▷

If the Port does not support 8.0 GT/s, this register is permitted to be hardwired to 0.

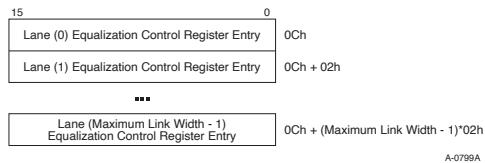


Figure 7-72 Lane Equalization Control Register §

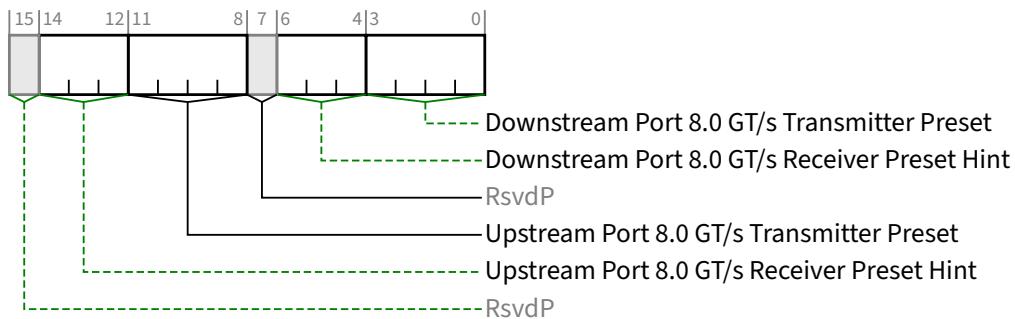


Figure 7-73 Lane Equalization Control Register Entry §

Table 7-60 Lane Equalization Control Register Entry §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|--|----------------------------------|---|----------------------------|-------|---|-----------------|-----|---|---|---------------|----|---|---|---------------|----|---|-------------------------------|
| 3:0 | <p>Downstream Port 8.0 GT/s Transmitter Preset - Transmitter preset value used for 8.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See § Chapter 8. for details. The field encodings are defined in § Section 4.2.4.2.</p> <p>For an Upstream Port if <u>Crosslink Supported</u> is 0b, this field is <u>RsvdP</u>. Otherwise, this field is <u>HwInit</u>. See § Section 7.5.3.18.</p> <p>The default value is 1111b.</p> | HwInit / RsvdP (see description) | | | | | | | | | | | | | | | | |
| 6:4 | <p>Downstream Port 8.0 GT/s Receiver Preset Hint - Receiver preset hint value that may be used as a suggested setting for 8.0 GT/s receiver equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See § Chapter 8. for details. The field encodings are defined in § Section 4.2.4.2.</p> <p>For an Upstream Port if <u>Crosslink Supported</u> is 0b, this field is <u>RsvdP</u>. Otherwise, this field is <u>HwInit</u>. See § Section 7.5.3.18.</p> <p>The default value is 111b.</p> | HwInit / RsvdP (see description) | | | | | | | | | | | | | | | | |
| 11:8 | <p>Upstream Port 8.0 GT/s Transmitter Preset - Field contains the Transmitter preset value sent or received during 8.0 GT/s Link Equalization. Field usage varies as follows:</p> <table border="1"> <thead> <tr> <th></th> <th>Operating Port Direction</th> <th><u>Crosslink Supported</u></th> <th>Usage</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>Downstream Port</td> <td>Any</td> <td> <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is <u>HwInit</u>.</p> </td></tr> <tr> <td>B</td> <td>Upstream Port</td> <td>0b</td> <td> <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value <i>MUST</i> be captured from EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is <u>RO</u>.</p> <p>Note: When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> </td></tr> <tr> <td>C</td> <td>Upstream Port</td> <td>1b</td> <td> <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is <u>HwInit</u>.</p> </td></tr> </tbody> </table> | | Operating Port Direction | <u>Crosslink Supported</u> | Usage | A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is <u>HwInit</u>.</p> | B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value <i>MUST</i> be captured from EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is <u>RO</u>.</p> <p>Note: When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is <u>HwInit</u>.</p> | HwInit / RO (see description) |
| | Operating Port Direction | <u>Crosslink Supported</u> | Usage | | | | | | | | | | | | | | | |
| A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is <u>HwInit</u>.</p> | | | | | | | | | | | | | | | |
| B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value <i>MUST</i> be captured from EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is <u>RO</u>.</p> <p>Note: When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | | | | | | | | | | | | | | | |
| C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is <u>HwInit</u>.</p> | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | | | Attributes |
|--------------|--|---------------------|---|--|
| | See § Section 4.2.4 and § Chapter 8 . for details. The field encodings are defined in § Section 4.2.4.2 . The default value is 1111b. | | | |
| 14:12 | Upstream Port 8.0 GT/s Receiver Preset Hint - Field contains the Receiver preset hint value sent or received during 8.0 GT/s Link Equalization. Field usage varies as follows: | | | HwInit / RO (see description) |
| | Operating Port Direction | Crosslink Supported | Usage | |
| A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit.</p> | |
| B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value <i>MUST</i>@FLIT be captured from EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO.</p> <p>Note: When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | |
| C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit.</p> | |
| | See § Section 4.2.4 and § Chapter 8 . for details. The field encodings are defined in § Section 4.2.4.2 . The default value is 111b. | | | |

7.7.4 Data Link Feature Extended Capability §

The Data Link Feature Capability is an optional Extended Capability that is required for Downstream Ports that support one or more of the associated features. Since the Scaled Flow Control Feature is required for Ports that support 16.0 GT/s, this capability is required for Downstream Ports that support 16.0 GT/s (see § [Section 3.4.2](#)). It is optional in other Downstream Ports. It is optional in Functions associated with an Upstream Port. In Multi-Function Devices associated with an Upstream Port, this capability is individually optional for each non-VF Function, and all implemented instances of this capability must report identical information in all fields of this capability. It is not applicable in Functions that are

not associated with a Port (e.g., RCiEPs, Root Complex Event Collectors). The Data Link Feature Extended Capability is shown in § Figure 7-74 .

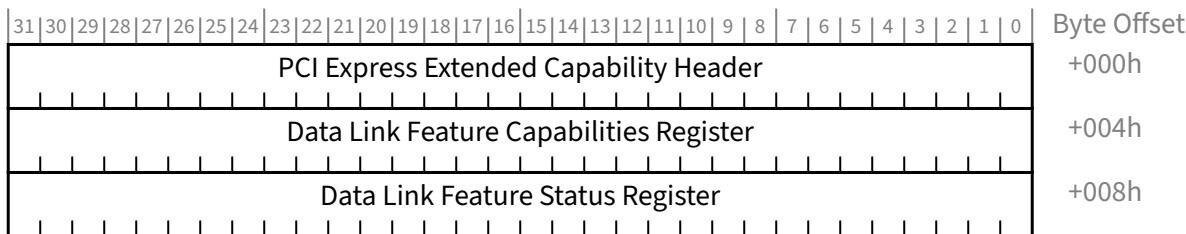


Figure 7-74 Data Link Feature Extended Capability §

7.7.4.1 Data Link Feature Extended Capability Header (Offset 00h) §

§ Figure 7-75 details allocation of register fields in the Data Link Feature Extended Capability Header ; § Table 7-63 provides the respective bit definitions.

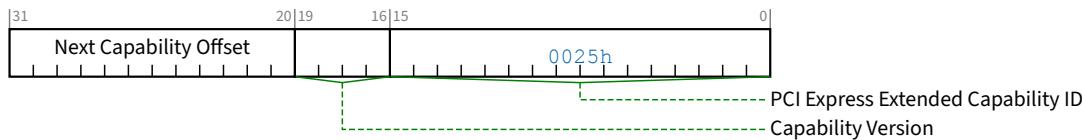


Figure 7-75 Data Link Feature Extended Capability Header §

Table 7-63 Data Link Feature Extended Capability Header §

| Bit Location | Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for Data Link Feature is 0025h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits. | RO |

7.7.4.2 Data Link Feature Capabilities Register (Offset 04h) §

§ Figure 7-76 details allocation of register fields in the Data Link Feature Capabilities register; § Table 7-64 provides the respective bit definitions.

When this Port sends a Data Link Feature DLLP , the Feature Support field in Symbols 1, 2, and 3 of that DLLP contains bits [22:16], [15:8], and [7:0] of this register respectively (See § Figure 3-14).

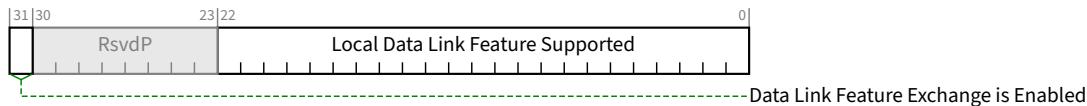


Figure 7-76 Data Link Feature Capabilities Register §

Table 7-64 Data Link Feature Capabilities Register §

| Bit Location | Description | Attributes |
|--------------|---|----------------|
| 22:0 | <p>Local Data Link Feature Supported - This field contains the Feature Supported value used when this Port sends a Data Link Feature DLLP (see § Figure 3-14). Defined features are:</p> <p>Bit 0 Local Scaled Flow Control Supported – Data Link Feature This bit indicates that this Port supports the Scaled Flow Control Feature (see § Section 3.4.2).</p> <p>Bit 1 Local Immediate Readiness – Data Link Parameter This bit indicates that all non-Virtual Functions in this Port have Immediate Readiness Set (see § Section 7.5.1.1.4). This bit MUST@FLIT be meaningful. In Non-Flit Mode, this bit is meaningful when Set, but when Clear indicates either that some non-Virtual Function has Immediate Readiness Clear or that this Port is not providing this information.</p> <p>Bits 4:2 Local Extended VC Count – Data Link Parameter This is the maximum number of Virtual Channels that can simultaneously be enabled, taking into account the Extended VC Count field in the Multi-Function Virtual Channel Extended Capability or the Virtual Channel Extended Capability (with Capability ID 0002h), and the SVC Extended VC Count field in the Streamlined Virtual Channel Extended Capability. This field is meaningful in Flit Mode. In Non-Flit Mode, this field must be zero.</p> <p>Bits 7:5 Local L0p Exit Latency – Data Link Parameter This field indicates this Port's knowledge of L0p Exit Latency. The value reported is the larger of Port L0p Exit Latency and Retimer L0p Exit Latency . The actual time required to widen the Link is the larger of Local L0p Exit Latency and Remote L0p Exit Latency . These values are a hint that approximates the typical exit latency. This value is used by the implementation specific L0p policy mechanism to help determine when it is reasonable to use L0p (and to what widths). The underlying implementation is permitted to take longer as permitted by § Section 4.2.6.7 . The Downstream Port's Retimer L0p Exit Latency should include retimers that are part of the “system”.</p> | HwInit / RsvdP |

| Bit Location | Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|--|-------------|----------------|-------------|----------------|-------------|----------------|-------------|----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|--|
| | <p>The Upstream Port's Retimer L0p Exit Latency should include retimers that are part of the “add-in card”.</p> <p>Defined encodings are:</p> <table> <tr><td>000b</td><td>Less than 1 µs</td></tr> <tr><td>001b</td><td>Less than 2 µs</td></tr> <tr><td>010b</td><td>Less than 4 µs</td></tr> <tr><td>011b</td><td>Less than 8 µs</td></tr> <tr><td>100b</td><td>Less than 16 µs</td></tr> <tr><td>101b</td><td>Less than 32 µs</td></tr> <tr><td>110b</td><td>Less than 64 µs</td></tr> <tr><td>111b</td><td>More than 64 µs</td></tr> </table> <p>This field is meaningful in Flit Mode. In Non-Flit Mode, this field is Reserved.</p> <p>Bits 22:8 RsvdP</p> <p>Other bits in this field are RsvdP .</p> | 000b | Less than 1 µs | 001b | Less than 2 µs | 010b | Less than 4 µs | 011b | Less than 8 µs | 100b | Less than 16 µs | 101b | Less than 32 µs | 110b | Less than 64 µs | 111b | More than 64 µs | |
| 000b | Less than 1 µs | | | | | | | | | | | | | | | | | |
| 001b | Less than 2 µs | | | | | | | | | | | | | | | | | |
| 010b | Less than 4 µs | | | | | | | | | | | | | | | | | |
| 011b | Less than 8 µs | | | | | | | | | | | | | | | | | |
| 100b | Less than 16 µs | | | | | | | | | | | | | | | | | |
| 101b | Less than 32 µs | | | | | | | | | | | | | | | | | |
| 110b | Less than 64 µs | | | | | | | | | | | | | | | | | |
| 111b | More than 64 µs | | | | | | | | | | | | | | | | | |
| 31 | <p>Data Link Feature Exchange is Enabled - If Set, indicates that this Port will enter the <u>DL_Feature</u> negotiation state (see § Section 3.2.1).</p> | HwInit | | | | | | | | | | | | | | | | |

7.7.4.3 Data Link Feature Status Register (Offset 08h) §

§ Figure 7-77 details allocation of register fields in the Data Link Feature Status Register ; § Table 7-65 provides the respective bit definitions.

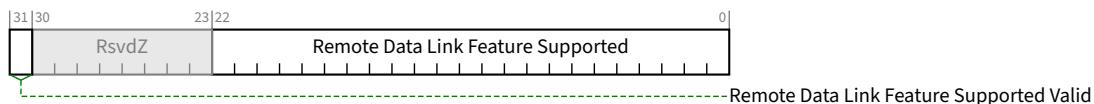


Figure 7-77 Data Link Feature Status Register §

Table 7-65 Data Link Feature Status Register §

| Bit Location | Description | Attributes |
|--------------|---|------------|
| 22:0 | <p>Remote Data Link Feature Supported - These bits indicate that the Remote Port supports the corresponding Data Link Feature. These bits capture all information from the Feature Supported field of the Data Link Feature DLLP even when this Port doesn't support the corresponding feature.</p> <p>This field is Cleared on entry to state <u>DL_Inactive</u> (see § Section 3.2.1).</p> <p>Features currently defined are:</p> <ul style="list-style-type: none"> Bit 0 Remote Scaled Flow Control Supported – Data Link Feature This bit indicates that the Remote Port supports the <u>Scaled Flow Control Feature</u> (see § Section 3.4.2). Bit 1 Remote Immediate Readiness – Data Link Parameter | RO |

Base 6.4 vs Base 6.3

| Bit Location | Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|--|-------------|----------------|-------------|----------------|-------------|----------------|-------------|----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|--|
| | <p>This bit indicates that all non-Virtual Functions in the Remote Port have <u>Immediate Readiness Set</u> (see § Section 7.5.1.1.4).</p> <p>In Flit Mode, this bit is always meaningful. In Non-Flit Mode, this bit is meaningful when Set, but when Clear indicates either that some non-Virtual Function has <u>Immediate Readiness Clear</u> or that the Remote Port is not providing this information.</p> <p>Bits 4:2 <i>Extended VC Count</i> – Data Link Parameter</p> <p>This is the maximum number of Virtual Channels that can simultaneously be enabled, taking into account the Extended VC Count field in the Multi-Function Virtual Channel Extended Capability or the Virtual Channel Extended Capability (with Capability ID 0002h), and the SVC Extended VC Count field in the Streamlined Virtual Channel Extended Capability .</p> <p>This field is meaningful in Flit Mode. In Non-Flit Mode, this field must be zero.</p> <p>Bits 7:5 <i>Remote L0p Exit Latency</i> – Data Link Parameter</p> <p>This field indicates the remote Port's L0p Exit Latency. The value reported indicates the length of time the remote Port requires to complete widening a link using L0p. If the remote Port does not support L0p, this field must contain 000b.</p> <p>These values are a hint that approximates the typical exit latency. This value is used by the implementation specific L0p policy mechanism to help determine when it is reasonable to use L0p (and to what widths). The underlying implementation is permitted to take longer as permitted by § Section 4.2.6.7 .</p> <p>Defined encodings are:</p> <table> <tr> <td>000b</td> <td>Less than 1 µs</td> </tr> <tr> <td>001b</td> <td>Less than 2 µs</td> </tr> <tr> <td>010b</td> <td>Less than 4 µs</td> </tr> <tr> <td>011b</td> <td>Less than 8 µs</td> </tr> <tr> <td>100b</td> <td>Less than 16 µs</td> </tr> <tr> <td>101b</td> <td>Less than 32 µs</td> </tr> <tr> <td>110b</td> <td>Less than 64 µs</td> </tr> <tr> <td>111b</td> <td>More than 64 µs</td> </tr> </table> <p>This field is meaningful in Flit Mode. In Non-Flit Mode, this field is Reserved.</p> <p>Bits 22:8 Reserved</p> <p>Default is 00 0000h</p> | 000b | Less than 1 µs | 001b | Less than 2 µs | 010b | Less than 4 µs | 011b | Less than 8 µs | 100b | Less than 16 µs | 101b | Less than 32 µs | 110b | Less than 64 µs | 111b | More than 64 µs | |
| 000b | Less than 1 µs | | | | | | | | | | | | | | | | | |
| 001b | Less than 2 µs | | | | | | | | | | | | | | | | | |
| 010b | Less than 4 µs | | | | | | | | | | | | | | | | | |
| 011b | Less than 8 µs | | | | | | | | | | | | | | | | | |
| 100b | Less than 16 µs | | | | | | | | | | | | | | | | | |
| 101b | Less than 32 µs | | | | | | | | | | | | | | | | | |
| 110b | Less than 64 µs | | | | | | | | | | | | | | | | | |
| 111b | More than 64 µs | | | | | | | | | | | | | | | | | |
| 31 | <p><i>Remote Data Link Feature Supported Valid</i> - This bit indicates that the Port has received a Data Link Feature DLLP in state DL_Feature (see § Section 3.2.1) and that the Remote Data Link Feature Supported field is meaningful. This bit is Cleared on entry to state DL_Inactive (see § Section 3.2.1).</p> <p>Default is 0b.</p> | RO | | | | | | | | | | | | | | | | |

7.7.5 Physical Layer 16.0 GT/s Extended Capability §

The Physical Layer 16.0 GT/s Extended Capability structure must be implemented in:

- A Function associated with a Downstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s.

- A Function of a Single-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s.
- Function 0 (and only Function 0) of a Multi-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s.

This capability is permitted to be implemented in any of the Functions listed above even if the 16.0 GT/s Link speed is not supported. Implementing this capability is strongly recommended for 8.0 GT/s only Flit Mode components. In Non-Flit Mode, when the 16.0 GT/s Link speed is not supported and in Flit Mode, when the 8.0 GT/s Link speed is not supported, the behavior of registers other than the Capability Header is undefined. In Flit Mode, operating at 8.0 GT/s, the Capability Header, 16.0 GT/s Local Data Parity Register , 16.0 GT/s First Retimer Data Parity Register , and 16.0 GT/s Second Retimer Data Parity Register are meaningful.

§ Figure 7-79 details allocation of register fields in the Physical Layer 16.0 GT/s Extended Capability structure.

Base 6.4 vs Base 6.3

| Byte Offset | 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|-------------|---|
| +000h | PCI Express Extended Capability Header |
| +004h | 16.0 GT/s Capabilities Register |
| +008h | 16.0 GT/s Control Register |
| +00Ch | 16.0 GT/s Status Register |
| +010h | 16.0 GT/s Local Data Parity Mismatch Status Register |
| +014h | 16.0 GT/s First Retimer Data Parity Mismatch Status Register |
| +018h | 16.0 GT/s Second Retimer Data Parity Mismatch Status Register |
| +01Ch | 16.0 GT/s Reserved |
| +020h | 16.0 GT/s Eq Ctl: Lane 3 16.0 GT/s Eq Ctl: Lane 2 16.0 GT/s Eq Ctl: Lane 1 16.0 GT/s Eq Ctl: Lane 0 |
| +024h | 16.0 GT/s Eq Ctl: Lane 7 16.0 GT/s Eq Ctl: Lane 6 16.0 GT/s Eq Ctl: Lane 5 16.0 GT/s Eq Ctl: Lane 4 |
| +028h | 16.0 GT/s Eq Ctl: Lane 11 16.0 GT/s Eq Ctl: Lane 10 16.0 GT/s Eq Ctl: Lane 9 16.0 GT/s Eq Ctl: Lane 8 |
| +02Ch | 16.0 GT/s Eq Ctl: Lane 15 16.0 GT/s Eq Ctl: Lane 14 16.0 GT/s Eq Ctl: Lane 13 16.0 GT/s Eq Ctl: Lane 12 |
| +030h | 16.0 GT/s Eq Ctl: Lane 19 16.0 GT/s Eq Ctl: Lane 18 16.0 GT/s Eq Ctl: Lane 17 16.0 GT/s Eq Ctl: Lane 16 |
| +034h | 16.0 GT/s Eq Ctl: Lane 23 16.0 GT/s Eq Ctl: Lane 22 16.0 GT/s Eq Ctl: Lane 21 16.0 GT/s Eq Ctl: Lane 20 |
| +038h | 16.0 GT/s Eq Ctl: Lane 27 16.0 GT/s Eq Ctl: Lane 26 16.0 GT/s Eq Ctl: Lane 25 16.0 GT/s Eq Ctl: Lane 24 |
| +03Ch | 16.0 GT/s Eq Ctl: Lane 31 16.0 GT/s Eq Ctl: Lane 30 16.0 GT/s Eq Ctl: Lane 29 16.0 GT/s Eq Ctl: Lane 28 |

Figure 7-78 Physical Layer 16.0 GT/s Extended Capability §

7.7.5.1 Physical Layer 16.0 GT/s Extended Capability Header (Offset 00h) §

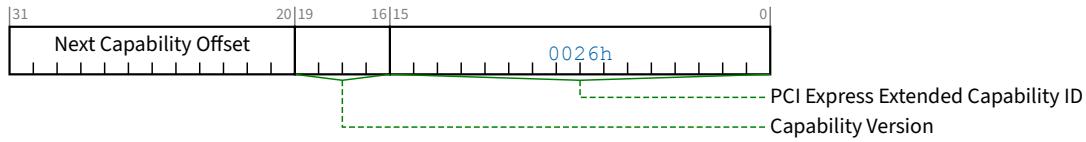


Figure 7-79 Physical Layer 16.0 GT/s Extended Capability Header §

Table 7-66 Physical Layer 16.0 GT/s Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Physical Layer 16.0 GT/s Capability is 0026h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.7.5.2 16.0 GT/s Capabilities Register (Offset 04h) §



Figure 7-80 16.0 GT/s Capabilities Register §

Table 7-67 16.0 GT/s Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|----------------------|------------|
| 31:0 | RsvdP RsvdP | RsvdP |

7.7.5.3 16.0 GT/s Control Register (Offset 08h) §

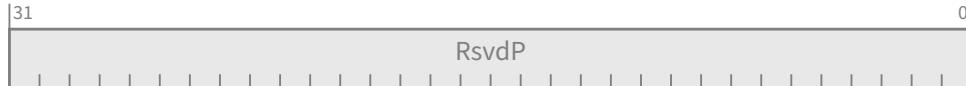


Figure 7-81 16.0 GT/s Control Register §

Table 7-68 16.0 GT/s Control Register §

| Bit Location | Register Description | Attributes |
|--------------|----------------------|------------|
| 31:0 | RsvdP RsvdP | RsvdP |

7.7.5.4 16.0 GT/s Status Register (Offset 0Ch) §

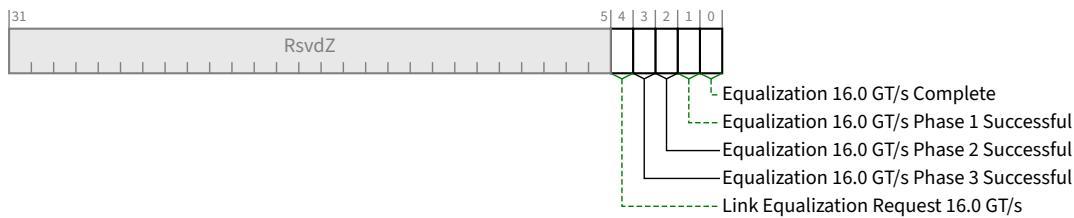


Figure 7-82 16.0 GT/s Status Register §

Table 7-69 16.0 GT/s Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|-------------|
| 0 | <p>Equalization 16.0 GT/s Complete - When Set, this bit indicates that the 16.0 GT/s Transmitter Equalization procedure has completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 1 | <p>Equalization 16.0 GT/s Phase 1 Successful - When set to 1b, this bit indicates that Phase 1 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |

| Bit Location | Register Description | Attributes |
|--------------|---|------------------|
| 2 | <p>Equalization 16.0 GT/s Phase 2 Successful - When set to 1b, this bit indicates that Phase 2 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 3 | <p>Equalization 16.0 GT/s Phase 3 Successful - When set to 1b, this bit indicates that Phase 3 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 4 | <p>Link Equalization Request 16.0 GT/s - This bit is Set by hardware to request the 16.0 GT/s Link equalization process to be performed on the Link. Refer to § Section 4.2.4 and § Section 4.2.7.4.2 for details.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | RW1CS / RsvdZ |

7.7.5.5 16.0 GT/s Local Data Parity Mismatch Status Register (Offset 10h) §

The 16.0 GT/s Local Data Parity Mismatch Status Register is a 32-bit vector where each bit indicates if the local Receiver detected a Data Parity mismatch on the Lane with the corresponding Lane number. This Lane number is the ~~↑↓default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.~~ ~~↑↓Physical Lane Number .~~

 Errata: Base 6.3
B825△◀▷

This register collects parity errors for 16.0 GT/s and higher data rates as well as 8.0 GT/s data rate in Flit Mode. When tracking errors for a specific Link Speed, software should clear this register on speed changes.



Figure 7-83 16.0 GT/s Local Data Parity Mismatch Status Register §

Table 7-70 16.0 GT/s Local Data Parity Mismatch Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------------|
| 31:0 | <p>Local Data Parity Mismatch Status - Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. See § Section 4.2.8.2 for more information.</p> <p>The default value of each bit is 0b.</p> <p>For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are RsvdZ.</p> | RW1CS / RsvdZ |

7.7.5.6 16.0 GT/s First Retimer Data Parity Mismatch Status Register (Offset 14h) §

The 16.0 GT/s First Retimer Data Parity Mismatch Status register is a 32-bit vector where each bit indicates if the first Retimer of a Path (see [§ Section 4.3.2](#) for more information) detected a Data Parity mismatch on the Lane with the corresponding Lane number. This Lane number is the [default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.](#) [Physical Lane Number](#)

Errata: Base 6.3
B825△◀▷

This register collects parity errors for 16.0 GT/s and higher data rates as well as 8.0 GT/s data rate in Flit Mode. When tracking errors for a specific Link Speed, software should clear this register on speed changes.

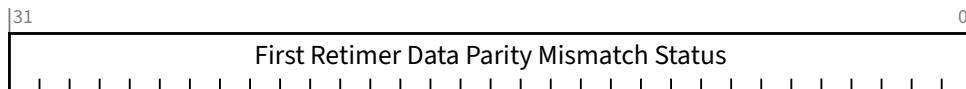


Figure 7-84 16.0 GT/s First Retimer Data Parity Mismatch Status Register §

Table 7-71 16.0 GT/s First Retimer Data Parity Mismatch Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------------|
| 31:0 | <p>First Retimer Data Parity Mismatch Status - Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. See § Section 4.2.8.2 for more information.</p> <p>The default value of each bit is 0b.</p> <p>The value of this field is undefined when no Retimers are present.</p> <p>For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are RsvdZ .</p> | RW1CS / RsvdZ |

7.7.5.7 16.0 GT/s Second Retimer Data Parity Mismatch Status Register (Offset 18h) §

The 16.0 GT/s Second Retimer Data Parity Mismatch Status Register is a 32-bit vector where each bit indicates if the second Retimer of a Path (see [§ Section 4.3.2](#) for more information) detected a Data Parity mismatch on the Lane with the corresponding Lane number. This Lane number is the [default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.](#) [Physical Lane Number](#)

Errata: Base 6.3
B825△◀▷

This register collects parity errors for 16.0 GT/s and higher data rates as well as 8.0 GT/s data rate in Flit Mode. When tracking errors for a specific Link Speed, software should clear this register on speed changes.



Figure 7-85 16.0 GT/s Second Retimer Data Parity Mismatch Status Register §

Table 7-72 16.0 GT/s Second Retimer Data Parity Mismatch Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|-------------------------|
| 31:0 | <p>Second Retimer Data Parity Mismatch Status - Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. See § Section 4.2.8.2 for more information.</p> <p>The default value of each bit is 0b.</p> <p>The value of this field is undefined when no Retimers are present or only one Retimer is present.</p> <p>For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are <u>RsvdZ</u>.</p> | RW1CS / <u>RsvdZ</u> |

7.7.5.8 Physical Layer 16.0 GT/s Reserved (Offset 1Ch) §

This register is RsvdP .

7.7.5.9 16.0 GT/s Lane Equalization Control Register (Offsets 20h to 3Ch) §

The Equalization Control register consists of control fields required for per-Lane 16.0 GT/s equalization. It contains entries for at least the number of Lanes defined by the Maximum Link Width (see § Section 7.5.3.6 or § Section 7.9.9.2), must be implemented in whole DW granularity (e.g., if the Maximum Link Width is x1, the register will still contain entries for 4 Lanes with the entries for Lanes 1, 2 and 3 being undefined), and it is permitted to contain up to 32 entries regardless of the Maximum Link Width . The value of entries beyond the Maximum Link Width is undefined.

Each entry contains the values for the Lane with the corresponding ↑↓ default Lane number which
is invariant to Link width and Lane reversal negotiation that occurs during Link training.↓
↑↓Physical Lane Number .↑

Errata: Base 6.3
B825△◀▷

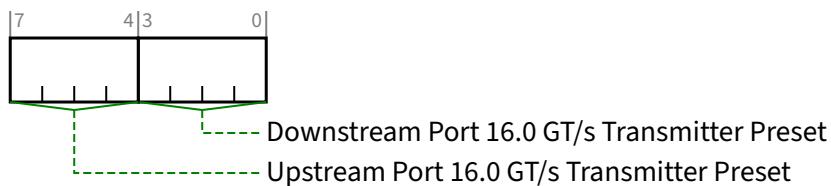


Figure 7-86 16.0 GT/s Lane Equalization Control Register Entry §

Table 7-73 16.0 GT/s Lane Equalization Control Register Entry §

| Bit Location | Register Description | Attributes |
|--------------|--|--|
| 3:0 | <p>Downstream Port 16.0 GT/s Transmitter Preset - Transmitter Preset used for 16.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See § Chapter 8. for details. The field encodings are defined in § Section 4.2.4.2 .</p> <p>For an Upstream Port if Crosslink Supported is 0b, this field is <u>RsvdP</u> . Otherwise, this field is <u>HwInit</u> . See § Section 7.5.3.18 .</p> <p>The default value is 1111b.</p> | HwInit / <u>RsvdP</u> (see description) |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | | | Attributes |
|--|--|---------------------|--|----------------------------------|
| 7:4 | Upstream Port 16.0 GT/s Transmitter Preset - Field contains the Transmit Preset value sent or received during 16.0 GT/s Link Equalization. Field usage varies as follows: | | | HwInit / RO (see description) |
| | Operating Port Direction | Crosslink Supported | Usage | |
| A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is <u>HwInit</u>.</p> | |
| B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value <i>MUST</i>@FLIT be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is <u>RO</u>.</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | |
| C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is <u>HwInit</u>.</p> | |
| <p>See <u>§ Section 4.2.4</u> and <u>§ Chapter 8</u>. for details. The field encodings are defined in <u>§ Section 4.2.4.2</u>.</p> <p>The default value is 1111b.</p> | | | | |

7.7.6 Physical Layer 32.0 GT/s Extended Capability §

The Physical Layer 32.0 GT/s Extended Capability structure must be implemented in Ports where one or more of the following features are supported:

- The Supported Link Speeds Vector field indicates support for a Link speed of 32.0 GT/s.
- The Function supports sending and/or receiving Modified TS1/TS2 Ordered Sets.

When implemented, this structure must be implemented in:

- A Function associated with a Downstream Port
- A Function of a Single-Function Device associated with an Upstream Port

- Function 0 (and only Function 0) of a Multi-Function Device associated with an Upstream Port

This capability is permitted to be implemented in any of the Functions listed above even if the 32.0 GT/s Link speed is not supported. When the 32.0 GT/s Link speed is not supported, the behavior of registers other than the Capability Header is undefined.

§ Figure 7-87 details allocation of register fields in the Physical Layer 32.0 GT/s Extended Capability structure.

Note that parity errors for 32.0 GT/s are recorded in 16.0 GT/s Local Data Parity Mismatch Status Register, 16.0 GT/s First Retimer Data Parity Mismatch Status Register, and 16.0 GT/s Second Retimer Data Parity Mismatch Status Register. When tracking errors for a specific Link Speed, software should clear those registers on speed changes.

Base 6.4 vs Base 6.3

| Byte Offset | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|---|---------------------------|---------------------------|---------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| +000h | PCI Express Extended Capability Header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +004h | 32.0 GT/s Capabilities Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +008h | 32.0 GT/s Control Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +00Ch | 32.0 GT/s Status Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +010h | Received Modified TS Data 1 Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +014h | Received Modified TS Data 2 Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +018h | Transmitted Modified TS Data 1 Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +01Ch | Transmitted Modified TS Data 2 Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +020h | 32.0 GT/s Eq Ctl: Lane 3 | 32.0 GT/s Eq Ctl: Lane 2 | 32.0 GT/s Eq Ctl: Lane 1 | 32.0 GT/s Eq Ctl: Lane 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +024h | 32.0 GT/s Eq Ctl: Lane 7 | 32.0 GT/s Eq Ctl: Lane 6 | 32.0 GT/s Eq Ctl: Lane 5 | 32.0 GT/s Eq Ctl: Lane 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +028h | 32.0 GT/s Eq Ctl: Lane 11 | 32.0 GT/s Eq Ctl: Lane 10 | 32.0 GT/s Eq Ctl: Lane 9 | 32.0 GT/s Eq Ctl: Lane 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +02Ch | 32.0 GT/s Eq Ctl: Lane 15 | 32.0 GT/s Eq Ctl: Lane 14 | 32.0 GT/s Eq Ctl: Lane 13 | 32.0 GT/s Eq Ctl: Lane 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +030h | 32.0 GT/s Eq Ctl: Lane 19 | 32.0 GT/s Eq Ctl: Lane 18 | 32.0 GT/s Eq Ctl: Lane 17 | 32.0 GT/s Eq Ctl: Lane 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +034h | 32.0 GT/s Eq Ctl: Lane 23 | 32.0 GT/s Eq Ctl: Lane 22 | 32.0 GT/s Eq Ctl: Lane 21 | 32.0 GT/s Eq Ctl: Lane 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +038h | 32.0 GT/s Eq Ctl: Lane 27 | 32.0 GT/s Eq Ctl: Lane 26 | 32.0 GT/s Eq Ctl: Lane 25 | 32.0 GT/s Eq Ctl: Lane 24 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +03Ch | 32.0 GT/s Eq Ctl: Lane 31 | 32.0 GT/s Eq Ctl: Lane 30 | 32.0 GT/s Eq Ctl: Lane 29 | 32.0 GT/s Eq Ctl: Lane 28 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Figure 7-87 Physical Layer 32.0 GT/s Extended Capability §

7.7.6.1 Physical Layer 32.0 GT/s Extended Capability Header (Offset 00h) §

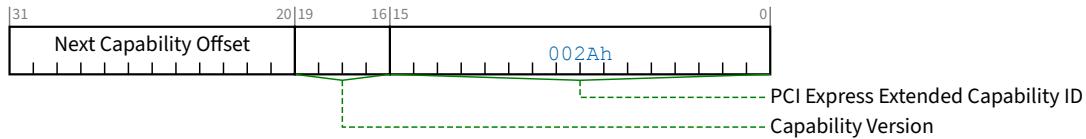


Figure 7-88 Physical Layer 32.0 GT/s Extended Capability Header §

Table 7-75 Physical Layer 32.0 GT/s Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Physical Layer 32.0 GT/s Capability is 002Ah . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.7.6.2 32.0 GT/s Capabilities Register (Offset 04h) §

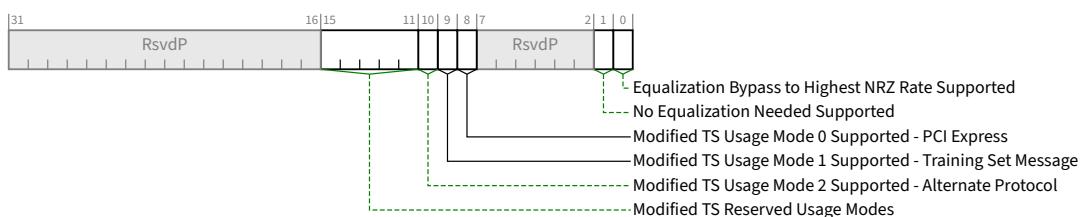


Figure 7-89 32.0 GT/s Capabilities Register §

Table 7-76 32.0 GT/s Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Equalization Bypass to Highest NRZ Rate Supported - When Set, this Port supports controlling whether the Port negotiates to skip equalization for speeds other than the highest common supported speed. See § Section 4.2.4 for details. Must be 1b for Ports that support 32.0 GT/s or higher data rates. | HwInit |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1 | No Equalization Needed Supported - When Set, this Port supports controlling whether or not Equalization is needed. | HwInit |
| 8 | Modified TS Usage Mode 0 Supported - PCI Express - This bit indicates that this Port supports PCI Express (Modified TS Usage 000b). This bit must be 1b. | RO |
| 9 | Modified TS Usage Mode 1 Supported - Training Set Message - This bit indicates that this Port supports sending and receiving vendor specific Training Set Messages (Modified TS Usage 001b). See § Section 4.2.5.2 for details. | HwInit |
| 10 | Modified TS Usage Mode 2 Supported - Alternate Protocol - This bit indicates that this Port supports negotiating to use alternate protocols (Modified TS Usage 010b). See § Section 4.2.5.2 for details. | HwInit |
| 15:11 | Modified TS Reserved Usage Modes - Reserved bits for future Usage Modes defined by the PCISIG. Must be 0 0000b. | RO |

7.7.6.3 32.0 GT/s Control Register (Offset 08h) §

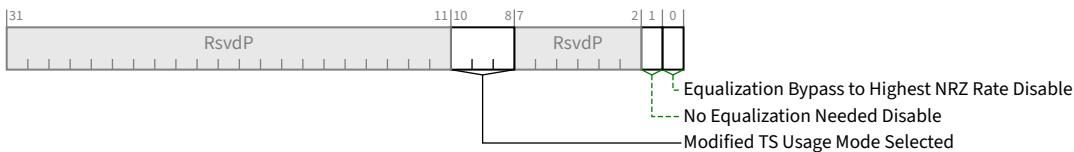


Figure 7-90 32.0 GT/s Control Register §

Table 7-77 32.0 GT/s Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------------|
| 0 | Equalization Bypass to Highest NRZ Rate Disable - When Clear, this Port is permitted to indicate during Link Training that it wishes to train to the highest common link NRZ data rate and skip equalization of intermediate data rates. See § Section 4.2.4 for details. If Equalization Bypass to Highest NRZ Rate Supported is Set, this bit is RWS with a default value of 0b. If Equalization Bypass to Highest NRZ Rate Supported is Clear, this bit is permitted to be hardwired to 0b. | RWS / RO |
| 1 | No Equalization Needed Disable - When Clear, this Port is permitted to indicate that it does not require equalization. When Set, this Port must always indicate that it requires equalization. See § Section 4.2.4 for details. If No Equalization Needed Supported is Set, this bit is RWS with a default value of 0b. If No Equalization Needed Supported is Clear, this bit is permitted to be hardwired to 0b. | RWS / RO |
| 10:8 | Modified TS Usage Mode Selected - This field indicates which Usage Mode will be used by this Downstream Port the next time the Link enters L0 LTSSM State. See § Section 4.2.5.2 for details. Behavior is undefined if this field indicates a Usage Mode that is not supported (i.e., associated Modified TS Usage Mode Supported bit is Clear). Unused bits in this field are permitted to be hardwired to 0b. If the only supported usage mode is PCI Express, this field is permitted to be hardwired to 000b. This field is present in Downstream Ports. In Upstream Ports, this field is RsvdP. Default is 000b. | RWS / RO / RsvdP |

7.7.6.4 32.0 GT/s Status Register (Offset 0Ch) §

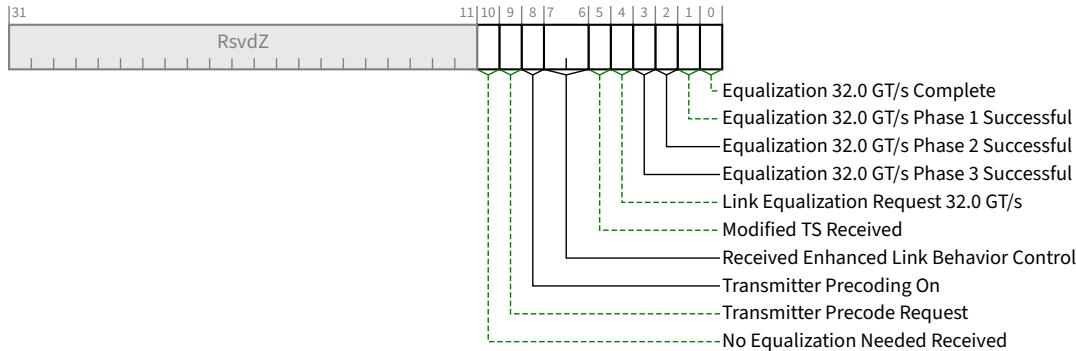


Figure 7-91 32.0 GT/s Status Register §

Table 7-78 32.0 GT/s Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 0 | <p>Equalization 32.0 GT/s Complete - When Set, this bit indicates that the 32.0 GT/s Transmitter Equalization procedure has completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 1 | <p>Equalization 32.0 GT/s Phase 1 Successful - When set to 1b, this bit indicates that Phase 1 of the 32.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 2 | <p>Equalization 32.0 GT/s Phase 2 Successful - When set to 1b, this bit indicates that Phase 2 of the 32.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 3 | <p>Equalization 32.0 GT/s Phase 3 Successful - When set to 1b, this bit indicates that Phase 3 of the 32.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| 4 | <p>Link Equalization Request 32.0 GT/s - This bit is Set by hardware to request the 32.0 GT/s Link equalization process to be performed on the Link. Refer to § Section 4.2.4 and § Section 4.2.7.4.2 for details.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | RW1CS / RsvdZ |
| 5 | <p>Modified TS Received - If Set, Received Modified TS Data 1 Register and Received Modified TS Data 2 Register contain meaningful data.</p> <p>This bit is Cleared when the Link is Down. This bit is Set when the <u>Modified TS1/TS2 Ordered Set</u> is received (See § Section 4.2.7.3.3). Default is 0b.</p> | RO |
| 7:6 | <p>Received Enhanced Link Behavior Control - This field contains the Enhanced Link Behavior Control bits from the most recent TS1 or TS2 received in the Polling or Configuration states. See § Section 4.2.5.1 , § Table 4-36 and § Table 4-37 .</p> <p>This field is Cleared on <u>DL_Down</u> .</p> <p>Default is 00b.</p> | RO |
| 8 | <p>Transmitter Precoding On - This field indicates whether the Receiver asked this transmitter to enable Precoding. See § Section 4.2.2.5 . This bit is cleared on <u>DL_Down</u> .</p> <p>Default is 0b.</p> | RO |
| 9 | <p>Transmitter Precode Request - When Set, this Port will request the transmitter to use Precoding by setting the Transmitter Precode Request bit in the TS1s/TS2s it transmits prior to entry to Recovery.Speed (see § Section 4.2.2.5).</p> <p>Default is Implementation Specific.</p> | RO |
| 10 | <p>No Equalization Needed Received - When Set, this Port either received a Modified TS1/TS2 with the No Equalization Needed bit Set or received a non-modified TS1/TS2 was received with the <u>No Equalization Needed</u> encoding (also reported in the Received Enhanced Link Behavior Control field).</p> <p>Default is 0b.</p> | RO |

7.7.6.5 Received Modified TS Data 1 Register (Offset 10h) §

This register contains the values received in the Modified TS1/TS2 Ordered Set (see § Table 4-38).

If PCI Express (Usage Mode 0) is the only one supported by a Port, this register is permitted to be hardwired to 0000 0000h.

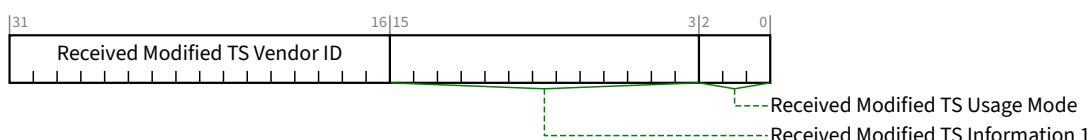


Figure 7-92 Received Modified TS Data 1 Register §

Table 7-79 Received Modified TS Data 1 Register §

| ↑↓Bit Location↓ ↑↓Bit Location↑ | Description | Attributes |
|------------------------------------|--|------------|
| 2:0 | <p>Received Modified TS Usage Mode - If Modified TS Received is Set, this field contains the Modified TS Usage field from the Modified TS1/TS2 Ordered Set (see § Section 4.2.7.3.6). If Modified TS Received is Clear, this field contains 000b.</p> <p>Unused bits in this field are permitted to be hardwired to 0b. If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 000b.</p> <p>Default is 000b.</p> | RO |
| 15:3 | <p>Received Modified TS Information 1 - If Modified TS Received is Set, this field contains the Modified TS Information 1 field from the Modified TS1/TS2 Ordered Set (see § Section 4.2.7.3.6). If Modified TS Received is Clear, this field contains 0 0000 0000 0000b.</p> <p>Bits 15:8 contain the value of Symbol 9.</p> <p>Bits 7:3 contain bits 7:3 of Symbol 8.</p> <p>If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 0 0000 0000 0000b.</p> <p>Default is 0 0000 0000 0000b.</p> | RO |
| 31:16 | <p>Received Modified TS Vendor ID - If Modified TS Received is Set, this field contains the Training Set Message Vendor ID or Alternate Protocol Vendor ID field from the Modified TS1/TS2 Ordered Set received (see § Section 4.2.7.3.6). If Modified TS Received is Clear, this field contains 0000h.</p> <p>Bits 15:8 contain the value of Symbol 11.</p> <p>Bits 7:0 contain the value of Symbol 10.</p> <p>If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 0000h.</p> <p>Default is 0000h.</p> | RO |

7.7.6.6 Received Modified TS Data 2 Register (Offset 14h) §

This register contains the values received in Symbols 12 through 14 of the Modified TS1/TS2 (see § Table 4-38).

If Modified TS Usage Mode 1 Supported - Training Set Message and Modified TS Usage Mode 2 Supported - Alternate Protocol are both Clear, this register is permitted to be hardwired to 0000 0000h.

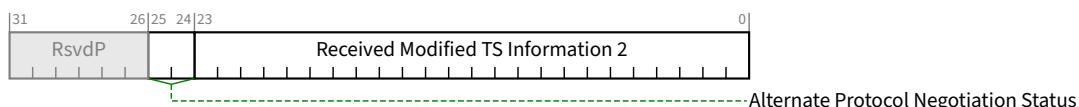


Figure 7-93 Received Modified TS Data 2 Register §

Table 7-80 Received Modified TS Data 2 Register §

| ↑↓Bit Location ↓ ↓↑Bit Location ↑ | Description | Attributes |
|--------------------------------------|--|------------|
| 23:0 | <p>Received Modified TS Information 2 - If Modified TS Received is Set, this field contains the Modified TS Information 2 field from the received Modified TS1/TS2 Ordered Set (§ Section 4.2.7.3.6). If Modified TS Received is Clear, this field contains 00 0000h.</p> <p>Bits 23:16 contain the value of Symbol 14.</p> <p>Bits 16:8 contain the value of Symbol 13.</p> <p>Bits 7:0 contain the value of Symbol 12.</p> <p>If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 00 0000h.</p> <p>Default is 00 0000h.</p> | RO |
| 25:24 | <p>Alternate Protocol Negotiation Status - Indicates the status of the Alternate Protocol Negotiation. Encodings are:</p> <ul style="list-style-type: none"> 00b Alternate Protocol Negotiation not supported 01b Alternate Protocol Negotiation disabled 10b Alternate Protocol Negotiation failed - Alternate Protocol Negotiation was attempted and did not locate a protocol that was supported on both ends of the Link. 11b Alternate Protocol Negotiation succeeded - Alternate Protocol Negotiation located one or more protocols that were supported on both ends of the Link and the Downstream Port selected one of those protocols for use. <p>If 11b, Alternate Protocol Negotiation completed successfully. If not 11b, Alternate Protocol Negotiation has not completed successfully. If Modified TS Usage Mode 1 Supported - Training Set Message and Modified TS Usage Mode 2 Supported - Alternate Protocol are both Clear, this field is permitted to be hardwired to 00b.</p> <p>If Modified TS Usage Mode 2 Supported - Alternate Protocol is Clear, this field is hardwired to 00b.</p> <p>If Modified TS Usage Mode 2 Supported - Alternate Protocol is Set and Modified TS Usage Mode Selected does not equal 2, this field must return a non-11b value. This field is cleared to 00b on entering Detect .</p> <p>Default is 00b.</p> | RO |

7.7.6.7 Transmitted Modified TS Data 1 Register (Offset 18h) §

This register contains the values transmitted in the Modified TS1/TS2 Ordered Set (see § Table 4-38).

If PCI Express (Usage Mode 0) is the only one supported by a Port, this register is permitted to be hardwired to 0000 0000h.

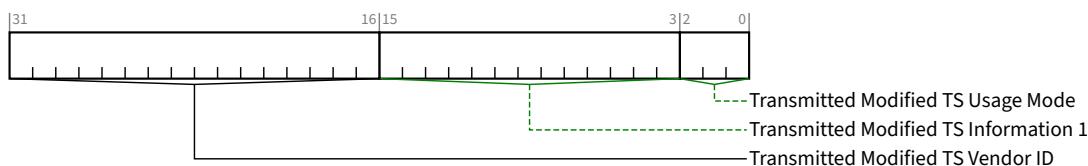


Figure 7-94 Transmitted Modified TS Data 1 Register §

Table 7-81 Transmitted Modified TS Data 1 Register §

| ↑↓Bit Location↓ ↑↓Bit Location↑ | Description | Attributes |
|------------------------------------|---|------------|
| 2:0 | <p>Transmitted Modified TS Usage Mode - If Modified TS Received is Set, this field contains the Modified TS Usage field from the Modified TS2 Ordered Set transmitted during the Configuration.Complete LTSSM State (see § Section 4.2.7.3.6).</p> <p>Unused bits in this field are permitted to be hardwired to 0b. If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 000b.</p> <p>Default is 000b.</p> | RO |
| 15:3 | <p>Transmitted Modified TS Information 1 - If Modified TS Received is Set, this field contains the Modified TS Information 1 field from Modified TS2 Ordered Set transmitted during the Configuration.Complete LTSSM State (see § Section 4.2.7.3.6).</p> <p>Bits 15:8 contain the value of Symbol 9.</p> <p>Bits 7:3 contain bits 7:3 of Symbol 8.</p> <p>If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 0 0000 0000 0000b.</p> <p>Default is 0 0000 0000 0000b.</p> | RO |
| 31:16 | <p>Transmitted Modified TS Vendor ID - If Modified TS Received is Set, this field contains the Training Set Message Vendor ID or Alternate Protocol Vendor ID field from the Modified TS2 Ordered Set transmitted during the Configuration.Complete LTSSM State (see § Section 4.2.7.3.6).</p> <p>Bits 15:8 contain the value of Symbol 11.</p> <p>Bits 7:0 contain the value of Symbol 10.</p> <p>If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 0000h.</p> <p>Default is 0000h.</p> | RO |

7.7.6.8 Transmitted Modified TS Data 2 Register (Offset 1Ch) §

This register contains the values received in Symbols 12 through 14 of the Modified TS1/TS2 (see § Table 4-38).

If Modified TS Usage Mode 1 Supported - Training Set Message and Modified TS Usage Mode 2 Supported - Alternate Protocol are both Clear, this register is permitted to be hardwired to 0000 0000h.

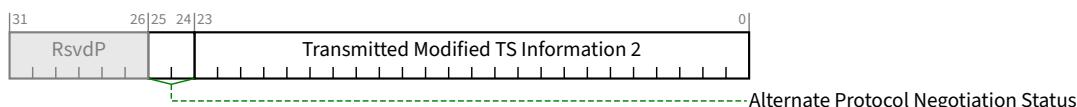


Figure 7-95 Transmitted Modified TS Data 2 Register §

Base 6.4 vs Base 6.3

Table 7-82 Transmitted Modified TS Data 2 Register §

| ↑↓Bit Location↓ ↓↑Bit Location↑ | Description | Attributes |
|------------------------------------|--|------------|
| 23:0 | <p>Transmitted Modified TS Information 2 - If Modified TS Received is Set, this field contains the Modified TS Information 2 field from the <u>Modified TS2 Ordered Set</u> transmitted during the Configuration.Complete LTSSM State (see § Section 4.2.7.3.6).</p> <p>Bits 23:16 contain the value of Symbol 14.</p> <p>Bits 16:8 contain the value of Symbol 13.</p> <p>Bits 7:0 contain the value of Symbol 12.</p> <p>If PCI Express (Usage Mode 0) is the only one supported, this field is permitted to be hardwired to 00 0000h.</p> <p>Default is 00 0000h.</p> | RO |
| 25:24 | <p>Alternate Protocol Negotiation Status - Indicates the status of the Alternate Protocol Negotiation. Encodings are:</p> <ul style="list-style-type: none"> 00b Alternate Protocol Negotiation not supported 01b Alternate Protocol Negotiation disabled 10b Alternate Protocol Negotiation failed - Alternate Protocol Negotiation was attempted and did not locate a protocol that was supported on both ends of the Link. 11b Alternate Protocol Negotiation succeeded - Alternate Protocol Negotiation located one or more protocols that were supported on both ends of the Link and the Downstream Port selected one of those protocols for use. <p>If 11b, Alternate Protocol Negotiation completed successfully. If not 11b, Alternate Protocol Negotiation has not completed successfully. If <u>Modified TS Usage Mode 1 Supported - Training Set Message</u> and <u>Modified TS Usage Mode 2 Supported - Alternate Protocol</u> are both Clear, this field is permitted to be hardwired to 00b.</p> <p>If <u>Modified TS Usage Mode 2 Supported - Alternate Protocol</u> is Clear, this field is hardwired to 00b.</p> <p>If <u>Modified TS Usage Mode 2 Supported - Alternate Protocol</u> is Set and <u>Modified TS Usage Mode Selected</u> does not equal 2, this field must return a non-11b value.</p> <p>This field is cleared to 00b on Detect.</p> <p>Default is 00b.</p> | RO |

7.7.6.9 32.0 GT/s Lane Equalization Control Register (Offset 20h) §

The 32.0 GT/s Equalization Control register consists of control fields required for per-Lane 32.0 GT/s equalization. It contains entries for at least the number of Lanes defined by the Maximum Link Width (see § Section 7.5.3.6 or § Section 7.9.9.2), must be implemented in whole DW granularity (e.g., if the Maximum Link Width is x1, the register will still contain entries for 4 Lanes with the entries for Lanes 1, 2 and 3 being undefined), and it is permitted to contain up to 32 entries regardless of the Maximum Link Width. The value of entries beyond the Maximum Link Width is undefined.

Each entry contains the values for the Lane with the corresponding ↑↓default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.↑
↑↓Physical Lane Number .↑

 Errata: Base 6.3
 B825△◀▶

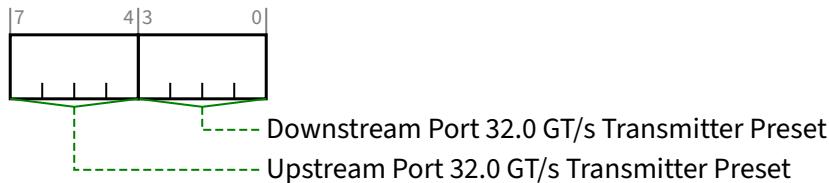


Figure 7-96 32.0 GT/s Lane Equalization Control Register Entry §

Table 7-83 32.0 GT/s Lane Equalization Control Register Entry §

| Bit Location | Register Description | | | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|---------------------|---|----------------------------------|--------------------------|---------------------|-------|---|-----------------|-----|---|---|---------------|----|---|---|---------------|----|---|-------------------------------|
| 3:0 | <p>Downstream Port 32.0 GT/s Transmitter Preset - Transmitter Preset used for 32.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See § Chapter 8. for details. The field encodings are defined in § Section 4.2.4.2.</p> <p>For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP . Otherwise, this field is HwInit . See § Section 7.5.3.18 .</p> <p>The default value is 1111b.</p> | | | HwInit / RsvdP (see description) | | | | | | | | | | | | | | | | |
| 7:4 | <p>Upstream Port 32.0 GT/s Transmitter Preset - Field contains the Transmit Preset value sent or received during 32.0 GT/s Link Equalization. Field usage varies as follows:</p> <table border="1"> <thead> <tr> <th></th> <th>Operating Port Direction</th> <th>Crosslink Supported</th> <th>Usage</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>Downstream Port</td> <td>Any</td> <td> <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit .</p> </td></tr> <tr> <td>B</td> <td>Upstream Port</td> <td>0b</td> <td> <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value MUST@FLIT be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO .</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> </td></tr> <tr> <td>C</td> <td>Upstream Port</td> <td>1b</td> <td> <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit .</p> </td></tr> </tbody> </table> <p>See § Section 4.2.4 and § Chapter 8. for details. The field encodings are defined in § Section 4.2.4.2 .</p> <p>The default value is 1111b.</p> | | | | Operating Port Direction | Crosslink Supported | Usage | A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit .</p> | B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value MUST@FLIT be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO .</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit .</p> | HwInit / RO (see description) |
| | Operating Port Direction | Crosslink Supported | Usage | | | | | | | | | | | | | | | | | |
| A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit .</p> | | | | | | | | | | | | | | | | | |
| B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value MUST@FLIT be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO .</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | | | | | | | | | | | | | | | | | |
| C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit .</p> | | | | | | | | | | | | | | | | | |

7.7.7 Physical Layer 64.0 GT/s Extended Capability §

The Physical Layer 64.0 GT/s Extended Capability structure must be implemented in Ports where one or more of the following features are supported:

- The Supported Link Speeds Vector field indicates support for a Link speed of 64.0 GT/s.

When implemented, this structure must be implemented in:

- A Function associated with a Downstream Port
- A Function of a Single-Function Device associated with an Upstream Port
- Function 0 (and only Function 0) of a Multi-Function Device associated with an Upstream Port

This capability is permitted to be implemented in any of the Functions listed above even if the 64.0 GT/s Link speed is not supported. When the 64.0 GT/s Link speed is not supported, the behavior of registers other than the Capability Header is undefined.

§ Figure 7-97 details allocation of register fields in the Physical Layer 64.0 GT/s Extended Capability structure.

Note that parity errors for 64.0 GT/s are recorded in 16.0 GT/s Local Data Parity Mismatch Status Register, 16.0 GT/s First Retimer Data Parity Mismatch Status Register, and 16.0 GT/s Second Retimer Data Parity Mismatch Status Register. When tracking errors for a specific Link Speed, software should clear those registers on speed changes.

| Byte Offset | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Physical Layer 64.0 GT/s Extended Capability Header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Capabilities Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Control Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Status Register | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64.0 GT/s Eq Ctl: Lane 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Figure 7-97 Physical Layer 64.0 GT/s Extended Capability §

7.7.7.1 Physical Layer 64.0 GT/s Extended Capability Header (Offset 00h) §

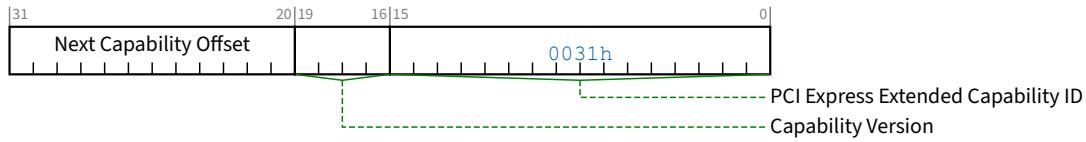


Figure 7-98 Physical Layer 64.0 GT/s Extended Capability Header §

Table 7-85 Physical Layer 64.0 GT/s Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | <p>PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Physical Layer 64.0 GT/s Capability is 0031h .</p> | RO |
| 19:16 | <p>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.</p> | RO |
| 31:20 | <p>Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.</p> | RO |

7.7.7.2 64.0 GT/s Capabilities Register (Offset 04h) §



Figure 7-99 64.0 GT/s Capabilities Register §

Table 7-86 64.0 GT/s Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|----------------------|------------|
| | | |

7.7.7.3 64.0 GT/s Control Register (Offset 08h) §

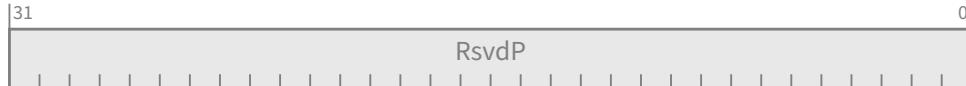


Figure 7-100 64.0 GT/s Control Register §

Table 7-87 64.0 GT/s Control Register §

| Bit Location | Register Description | Attributes |
|--------------|----------------------|------------|
|--------------|----------------------|------------|

7.7.7.4 64.0 GT/s Status Register (Offset 0Ch) §

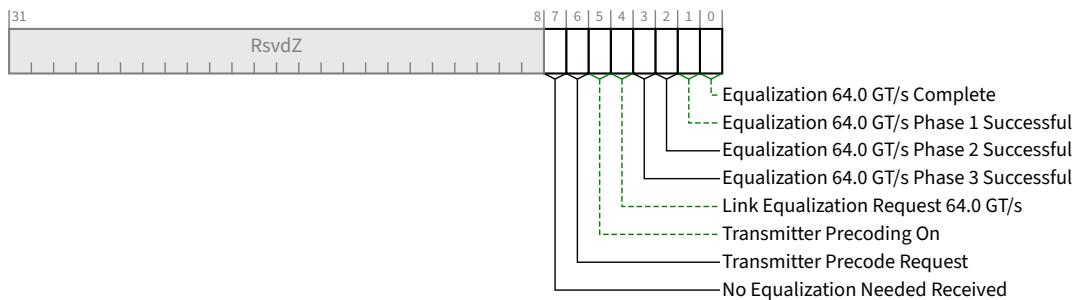


Figure 7-101 64.0 GT/s Status Register §

Table 7-88 64.0 GT/s Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|-------------|
| 0 | <p>Equalization 64.0 GT/s Complete - When Set, this bit indicates that the 64.0 GT/s Transmitter Equalization procedure has completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 1 | <p>Equalization 64.0 GT/s Phase 1 Successful - When set to 1b, this bit indicates that Phase 1 of the 64.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2 .</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| 2 | <p>Equalization 64.0 GT/s Phase 2 Successful - When set to 1b, this bit indicates that Phase 2 of the 64.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 3 | <p>Equalization 64.0 GT/s Phase 3 Successful - When set to 1b, this bit indicates that Phase 3 of the 64.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in § Section 4.2.7.4.2.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | ROS / RsvdZ |
| 4 | <p>Link Equalization Request 64.0 GT/s - This bit is Set by hardware to request the 64.0 GT/s Link equalization process to be performed on the Link. Refer to § Section 4.2.4 and § Section 4.2.7.4.2 for details.</p> <p>The default value of this bit is 0b.</p> <p>For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.</p> | RW1CS / RsvdZ |
| 5 | <p>Transmitter Precoding On - This field indicates whether the Receiver asked this transmitter to enable Precoding. See § Section 4.2.3.1.4. This bit is cleared on DL_Down.</p> <p>Default is 0b.</p> | RO |
| 6 | <p>Transmitter Precode Request - When Set, this Port will request the transmitter to use Precoding by setting the Transmitter Precode Request bit in the TS1s/TS2s it transmits prior to entry to Recovery.Speed (see § Section 4.2.3.1.4).</p> <p>Default is Implementation Specific.</p> | RO |
| 7 | <p>No Equalization Needed Received - When Set, this Port either received a Modified TS1/TS2 with the No Equalization Needed bit Set or received a non-modified TS1 / TS2 was received with the No Equalization Needed encoding (also reported in the Received Enhanced Link Behavior Control field).</p> <p>Default is 0b.</p> | RO |

7.7.7.5 64.0 GT/s Lane Equalization Control Register (Offset 10h) §

The 64.0 GT/s Equalization Control register consists of control fields required for per-Lane 64.0 GT/s equalization. It contains entries for at least the number of Lanes defined by the Maximum Link Width (see § [Section 7.5.3.6](#) or § [Section 7.9.9.2](#)), must be implemented in whole DW granularity (e.g., if the Maximum Link Width is x1, the register will still contain entries for 4 Lanes with the entries for Lanes 1, 2 and 3 being undefined), and it is permitted to contain up to 32 entries regardless of the Maximum Link Width. The value of entries beyond the Maximum Link Width is undefined.

Each entry contains the values for the Lane with the corresponding [↑↓ default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.↓](#)
[↑↓ Physical Lane Number .↑](#)

 Errata: Base 6.3
 B825△◀▷

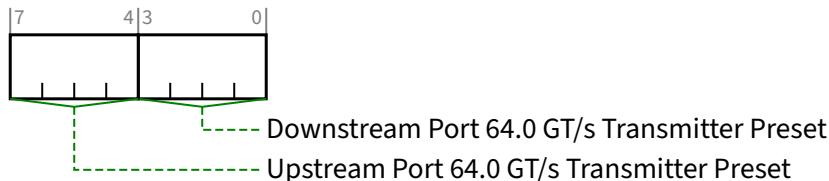


Figure 7-102 64.0 GT/s Lane Equalization Control Register Entry §

Table 7-89 64.0 GT/s Lane Equalization Control Register Entry §

| Bit Location | Register Description | | | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|---------------------|--|----------------------------------|--------------------------|---------------------|-------|---|-----------------|-----|---|---|---------------|----|--|---|---------------|----|---|-------------------------------|
| 3:0 | <p>Downstream Port 64.0 GT/s Transmitter Preset - Transmitter Preset used for 64.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See § Chapter 8. for details. The field encodings are defined in § Table 4-34.</p> <p>For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP . Otherwise, this field is HwInit . See § Section 7.5.3.18 .</p> <p>The default value is 1111b.</p> | | | HwInit / RsvdP (see description) | | | | | | | | | | | | | | | | |
| 7:4 | <p>Upstream Port 64.0 GT/s Transmitter Preset - Field contains the Transmit Preset value sent or received during 64.0 GT/s Link Equalization. Field usage varies as follows:</p> <table border="1"> <thead> <tr> <th></th> <th>Operating Port Direction</th> <th>Crosslink Supported</th> <th>Usage</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>Downstream Port</td> <td>Any</td> <td> <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit .</p> </td></tr> <tr> <td>B</td> <td>Upstream Port</td> <td>0b</td> <td> <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value must be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO .</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> </td></tr> <tr> <td>C</td> <td>Upstream Port</td> <td>1b</td> <td> <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit .</p> </td></tr> </tbody> </table> <p>See § Section 4.2.4 and § Chapter 8. for details. The field encodings are defined in § Table 4-34 .</p> <p>The default value is 1111b.</p> | | | | Operating Port Direction | Crosslink Supported | Usage | A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit .</p> | B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value must be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO .</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit .</p> | HwInit / RO (see description) |
| | Operating Port Direction | Crosslink Supported | Usage | | | | | | | | | | | | | | | | | |
| A | Downstream Port | Any | <p>Field contains the value sent on the associated Lane during Recovery.RcvrCfg.</p> <p>Field is HwInit .</p> | | | | | | | | | | | | | | | | | |
| B | Upstream Port | 0b | <p>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.</p> <p>This value must be captured from 128b/130b EQ TS2 or equalization requests with Use_Preset Set are received. This value should not be affected by equalization requests with Use_Preset Clear.</p> <p>Field is RO .</p> <p>When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</p> | | | | | | | | | | | | | | | | | |
| C | Upstream Port | 1b | <p>Field is not used or affected by the current Link Equalization.</p> <p>Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.</p> <p>Field is HwInit .</p> | | | | | | | | | | | | | | | | | |

7.7.8 Flit Logging Extended Capability §

This capability *MUST* be implemented in Ports and RCRBs that support Flit Mode. For Functions associated with an Upstream Port, this capability *MUST* be implemented in Function 0 and *MUST* not be implemented in any other Function of that Upstream Port.

This capability is only used in Flit Mode. The capability has no effect in Non-Flit Mode.

§ Figure 7-103 details allocation of the register bits in the Flit Logging Extended Capability structure.

The Lane numbers in FBER Measurement Status 3 Register through FBER Measurement Status 10 Register are strongly recommended to be the default Lane numbers which are invariant to Link width and Lane reversal negotiation that occurs during Link training. ↴↑Physical Lane Number.↑

Errata: Base 6.3
B825△◀▷

Base 6.4 vs Base 6.3

| Byte Offset | |
|-------------|---|
| +000h | 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
| +004h | Flit Logging Extended Capability Header |
| +008h | Flit Error Log 1 Register |
| +00Ch | Flit Error Log 2 Register |
| +010h | Flit Error Counter Status Register Flit Error Counter Control Register |
| +014h | FBER Measurement Control Register |
| +018h | FBER Measurement Status 1 Register |
| +01Ch | FBER Measurement Status 2 Register |
| +020h | FBER Measurement Status 3 Register |
| +024h | FBER Measurement Status 4 Register |
| +028h | FBER Measurement Status 5 Register |
| +02Ch | FBER Measurement Status 6 Register |
| +030h | FBER Measurement Status 7 Register |
| +034h | FBER Measurement Status 8 Register |
| +038h | FBER Measurement Status 9 Register |
| | FBER Measurement Status 10 Register |

Figure 7-103 Flit Logging Extended Capability Structure §

7.7.8.1 Flit Logging Extended Capability Header (Offset 00h) §

§ Figure 7-104 details allocation of the register fields in the Flit Logging Extended Capability Header ; § Table 7-91 provides the respective bit definitions.

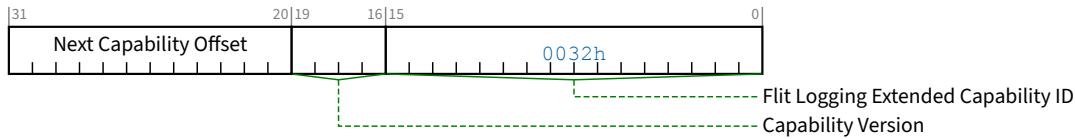


Figure 7-104 Flit Logging Extended Capability Header §

Table 7-91 Flit Logging Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Flit Logging Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Flit Logging Extended Capability is 0032h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.7.8.2 Flit Error Log 1 Register (Offset 04h) §

The Flit Error Log 1 Register and Flit Error Log 2 Register are Link level registers and contain information about the Flit errors corrected and/or detected by the FEC and/or CRC in a received Flit. The Flit Error Log is a FIFO of an implementation specific and unspecified size (size 1 is permitted). These registers contain the oldest log entry. Clearing the Flit Error Log Valid removes the oldest log entry from the FIFO and loads these registers with the next oldest log entry (if there is one). See § Section 4.2.3.1.2 , § Section 4.2.3.1.3 , § Appendix J. , and § Appendix K. for details.

Errors logged in the Flit Error Log 1 register and the Flit Error Log 2 register are interpreted as shown in Table 7-x.

Table 7-92 Flit Error Log Interpretation §

| Flit Error Log Valid | FEC Uncorrectable Error in Flit | Unrecognized Flit | Syndrome Parity and Check for ECC Groups 0, 1, and 2 | Flit Error Log Entry Interpretation | Notes |
|----------------------|---------------------------------|-------------------|--|-------------------------------------|--------|
| 1b | 0b | 0b | All 00h | Reserved | Note 1 |
| 1b | 0b | 0b | At least one != 00h | Good Flit, correctable error | |

| Flit Error Log Valid | FEC Uncorrectable Error in Flit | Unrecognized Flit | Syndrome Parity and Check for ECC Groups 0, 1, and 2 | Flit Error Log Entry Interpretation | Notes |
|----------------------|---------------------------------|-------------------|--|--|-------|
| 1b | 0b | 1b | All 00h | Unrecognized Flit with no accompanying FEC correctable error | |
| 1b | 0b | 1b | At least one != 00h | Unrecognized Flit with accompanying FEC correctable error | |
| 1b | 1b | x | All 00h | Uncorrectable error due to CRC, FEC OK | |
| 1b | 1b | x | At least one != 00h | Uncorrectable error due to FEC, CRC, or both | |
| 0b | x | x | Any | No Log Entry Present | |

Notes:

1. Software should silently discard this log entry

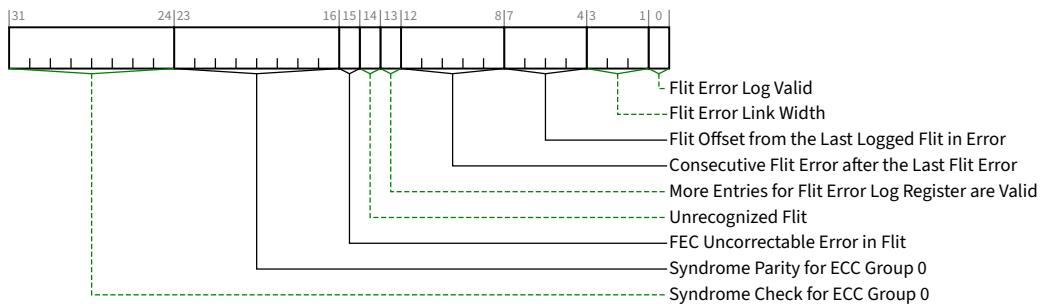


Figure 7-105 Flit Error Log 1 Register §

Table 7-93 Flit Error Log 1 Register §

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|--|-------------|----|-------------|----|-------------|----|-------------|----|-----|
| 0 | <p>Flit Error Log Valid – This bit is Set to 1b when an error is logged in this register and the <u>Flit Error Log 2 Register</u>.</p> <p>Writing 1b to this bit either clears this bit or, if <u>More Entries for Flit Error Log Register are Valid</u> (i.e., Bit 13) is Set, loads the <u>Flit Error Log 1 Register</u> and <u>Flit Error Log 2 Register</u> with the next oldest log entry.</p> <p>Default Zero .</p> | RW1CS | | | | | | | | |
| 3:1 | <p>Flit Error Link Width – Link Width when error was logged (taking into account any narrowing due to <u>L0p</u>). Encoding is:</p> <table> <tr> <td>000b</td> <td>x1</td> </tr> <tr> <td>001b</td> <td>x2</td> </tr> <tr> <td>010b</td> <td>x4</td> </tr> <tr> <td>011b</td> <td>x8</td> </tr> </table> | 000b | x1 | 001b | x2 | 010b | x4 | 011b | x8 | ROS |
| 000b | x1 | | | | | | | | | |
| 001b | x2 | | | | | | | | | |
| 010b | x4 | | | | | | | | | |
| 011b | x8 | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>100b x16</p> <p>Others Reserved</p> <p>Default is <u>Zero</u>.</p> | |
| 7:4 | <p>Flit Offset from the Last Logged Flit in Error – This is the offset from the last Flit whose error has been recorded in the prior entry of the Flit Error Log Register, if any.</p> <ul style="list-style-type: none"> • If this is the very first error that gets logged or this is the only copy of the Flit Error Log Register, this value must be 0h. • If the previous Flit was in error and logged, this value must be 1h. • If the last logged Flit was more than 15 Flits away, this value must be Fh. <p>This field only reflects errors that were logged. If the previous Flit was in error but was not logged, that error has no effect on this value.</p> <p>Default is <u>Zero</u>.</p> | ROS |
| 12:8 | <p>Consecutive Flit Error after the Last Flit Error – Initially, this field is <u>Zero</u>. If there are any errors (either correctable or uncorrectable by FEC) in any of the 5 consecutive Flits immediately following the Flit recorded in this log entry, the corresponding bits are set to 1b; otherwise they remain 0b. This field can change value after <u>Flit Error Log Valid</u> is Set and more Flits are received. If <u>More Entries for Flit Error Log Register</u> is Set, some bits of this field may not be meaningful and software can determine the accurate complete Flit error status from this field and subsequent log entries using the following example as a model.</p> <p>Consider consecutive log entries A, B and C, where A is older than B and B is older than C:</p> <ul style="list-style-type: none"> • If <u>Flit Offset from the Last Logged Flit in Error</u> in B is >5, then this field in log entry A is accurate (since there were more than 5 intervening Flits between A and B). • If <u>Flit Offset from the Last Logged Flit in Error</u> in B is ≤ 5, then some bits of this field in A must be determined by software using B (and C, if applicable, depending on the distance between A and C). • If <u>Flit Offset from the Last Logged Flit in Error</u> in B = 2, then entry A is for two Flits earlier. For entry A: <ul style="list-style-type: none"> ◦ Bit 0 represents the intervening Flit (which might have been in error but not logged) and ◦ Bits 4:1 are not meaningful and must be determined by software using B (and C, if applicable, depending on the distance between A and C). Bit 1 is 1b (because B exists), and bits 4:2 are bits 2:0 of B. ◦ If in turn, <u>Flit Offset from the Last Logged Flit in Error</u> in C is ≤5, some of the bits in B are not meaningful and must be determined by software using C (and possibly the next entry D, if applicable and available). <p>Default is <u>Zero</u>.</p> | ROS |
| 13 | <p>More Entries for Flit Error Log Register are Valid – when Set, it indicates that the Port has additional valid copies of the Flit Error Log Register for subsequent Flits. A port that implements only one set of Flit Error Log Register is permitted to hardwire this to <u>Zero</u>.</p> <p>If this bit is Set, clearing the Flit Error Log Valid bit loads the next oldest Flit Error Log Register entry. This bit can change value when <u>Flit Error Log Valid</u> bit is Set and an additional error is being logged.</p> <p>Default is <u>Zero</u>.</p> | ROS |
| 14 | <p>Unrecognized Flit – when Set indicates receipt of a Flit that passes CRC after FEC decode but uses a Reserved encoding in the <u>Flit Usage</u> or <u>Flit_Status</u> fields.</p> | ROS |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | Default is Zero | |
| 15 | FEC Uncorrectable Error in Flit – When set to 1b indicates either a CRC mismatch or one of the three FEC groups detecting an error it could not correct | <u>ROS</u> |
| 23:16 | Syndrome Parity for ECC Group 0 – Synd_Parity in § Chapter 4.. Default is Zero . | <u>ROS</u> |
| 31:24 | Syndrome Check for ECC Group 0 – Synd_Check in § Chapter 4.. Default is Zero . | <u>ROS</u> |

7.7.8.3 Flit Error Log 2 Register (Offset 08h) §

The Flit Error Log 1 Register and Flit Error Log 2 Register are Link level registers and contain information about the Flit errors corrected and/or detected by the FEC and/or CRC in a received Flit.

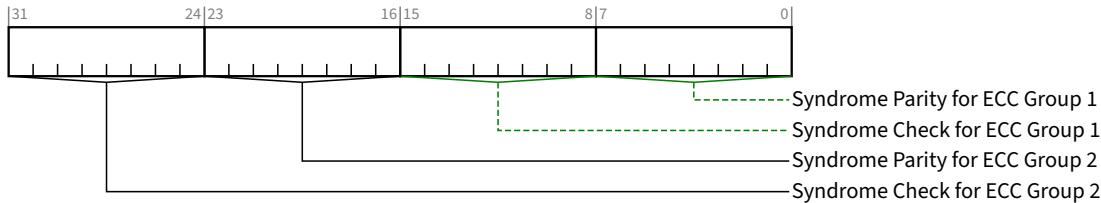


Figure 7-106 Flit Error Log 2 Register §

Table 7-94 Flit Error Log 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Syndrome Parity for ECC Group 1 – Synd_Parity in § Chapter 4.. Default is Zero . | <u>ROS</u> |
| 15:8 | Syndrome Check for ECC Group 1 – Synd_Check in § Chapter 4.. Default is Zero . | <u>ROS</u> |
| 23:16 | Syndrome Parity for ECC Group 2 – Synd_Parity in § Chapter 4.. Default is Zero . | <u>ROS</u> |
| 31:24 | Syndrome Check for ECC Group 2 – Synd_Check in § Chapter 4.. Default is Zero . | <u>ROS</u> |

7.7.8.4 Flit Error Counter Control Register (Offset 0Ch) §

The Flit Error Counter registers are Link wide and count the number of Flit and/or Ordered Set errors occurring on a Link operating in Flit Mode.

Base 6.4 vs Base 6.3

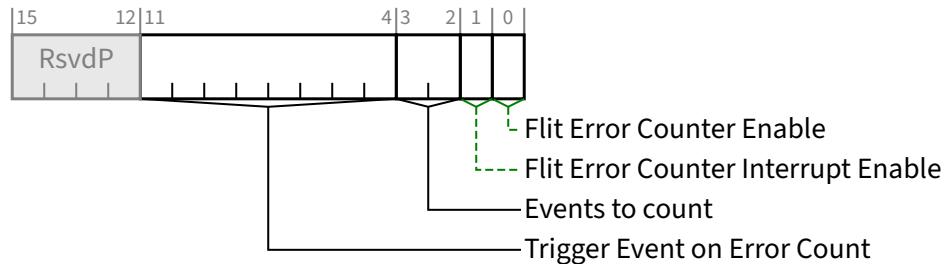


Figure 7-107 Flit Error Counter Control Register §

Table 7-95 Flit Error Counter Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Flit Error Counter Enable – Setting this bit enables and starts the Flit Error Counter in the Link. Clearing this bit stops the Flit Error Counter from incrementing. When Clear, it is strongly recommended that Flit Error Counter does not decrement.</p> <p>Default is Zero .</p> | RW |
| 1 | <p>Flit Error Counter Interrupt Enable – Generate an interrupt when Interrupt Generated based on Trigger Event Count transitions from 0b to 1b.</p> <p>The interrupt vector is Interrupt Message Number (see § Section 7.5.3.2).</p> <p>Default is Zero.</p> | RW |
| 3:2 | <p>Events to count –</p> <ul style="list-style-type: none"> 00b FEC-correctable Flit (see § Section 4.2.3.4.2), Invalid Flit (see § Section 4.2.3.4.2), or Framing Error (see § Section 4.2.2.3.4 and § Section 4.2.3.2) 01b FEC-correctable Flit 10b Invalid Flit 11b All events in 00b plus: <ul style="list-style-type: none"> • a 1b/1b TS Ordered Set on any Lane with only one valid half (see § Section 4.2.5.1) • an invalid Ordered Set on any Lane <p>Default is Zero .</p> | RW |
| 11:4 | <p>Trigger Event on Error Count – Generate an event (interrupt, if enabled) if this field is non-zero and the Flit Error Counter field in Flit Error Counter Status Register exceeds this value.</p> <p>Default is Zero .</p> | RW |

7.7.8.5 Flit Error Counter Status Register (Offset 0Eh) §

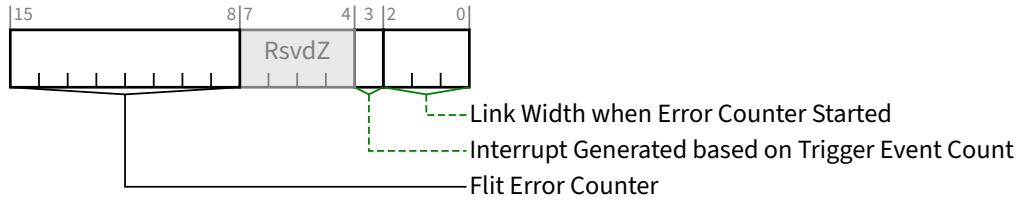


Figure 7-108 Flit Error Counter Status Register §

Table 7-96 Flit Error Counter Status Register §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | |
|------------------|--|--------------|--|---------------|--|------------------|--|-------------|----|-------------|-----|---------------|----------|------------|
| 2:0 | <p>Link Width when Error Counter Started – This field tracks the link width when the error counter started or restarted counting. The encodings are as follows:</p> <table> <tr><td>000b</td><td>x1</td></tr> <tr><td>001b</td><td>x2</td></tr> <tr><td>010b</td><td>x4</td></tr> <tr><td>011b</td><td>x8</td></tr> <tr><td>100b</td><td>x16</td></tr> <tr><td>Others</td><td>Reserved</td></tr> </table> <p>Default is Zero .</p> | 000b | x1 | 001b | x2 | 010b | x4 | 011b | x8 | 100b | x16 | Others | Reserved | <u>ROS</u> |
| 000b | x1 | | | | | | | | | | | | | |
| 001b | x2 | | | | | | | | | | | | | |
| 010b | x4 | | | | | | | | | | | | | |
| 011b | x8 | | | | | | | | | | | | | |
| 100b | x16 | | | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | | | |
| 3 | <p>Interrupt Generated based on Trigger Event Count – hardware Sets this bit when an interrupt condition is generated based on the trigger event. While this bit is Set, no new interrupts will be generated based on the trigger event.</p> <p>Cleared on 0b to 1b transition of Flit Error Counter Enable .</p> <p>It is strongly recommended that software transition Flit Error Counter Enable from 0b to 1b after the interrupt is serviced.</p> <p>Default is Zero .</p> | <u>RW1CS</u> | | | | | | | | | | | | |
| 15:8 | <p>Flit Error Counter – Increments by 1 when enabled and a countable event has occurred, as defined in the Flit Error Counter Control Register .</p> <p>Decrement by 1 at a fixed rate, if non-zero, based on Encoding and Link Width as follows:</p> <table> <tr><td>1b/1b</td><td>$\frac{10^6}{(\text{Link Width} \times 2)}$ UI (± 5 ns)</td></tr> <tr><td>8b/10b</td><td>$\frac{10^{12}}{\text{Link Width}}$ UI (± 5 ns)</td></tr> <tr><td>128b/130b</td><td>$\frac{10^{12}}{\text{Link Width}}$ UI (± 5 ns)</td></tr> </table> <p>Cleared on 0b to 1b transition of Flit Error Counter Enable .</p> <p>Does not roll over.</p> <p>Default is Zero .</p> | 1b/1b | $\frac{10^6}{(\text{Link Width} \times 2)}$ UI (± 5 ns) | 8b/10b | $\frac{10^{12}}{\text{Link Width}}$ UI (± 5 ns) | 128b/130b | $\frac{10^{12}}{\text{Link Width}}$ UI (± 5 ns) | <u>ROS</u> | | | | | | |
| 1b/1b | $\frac{10^6}{(\text{Link Width} \times 2)}$ UI (± 5 ns) | | | | | | | | | | | | | |
| 8b/10b | $\frac{10^{12}}{\text{Link Width}}$ UI (± 5 ns) | | | | | | | | | | | | | |
| 128b/130b | $\frac{10^{12}}{\text{Link Width}}$ UI (± 5 ns) | | | | | | | | | | | | | |

7.7.8.6 FBER Measurement Control Register (Offset 10h) §

The FBER Measurement Control register enables direct FBER measurement with status reported in the FBER Measurement Status Registers. For Retimers, the control is provided through Margin Command in the Control SKP Ordered Set from the Downstream Port (see § Chapter 4.).

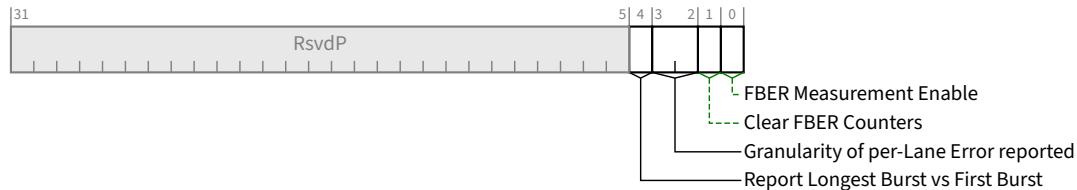


Figure 7-109 FBER Measurement Control Register §

Table 7-97 FBER Measurement Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | FBER Measurement Enable – Setting this bit enables and starts the FBER measurement in the Link. Clearing this bit stops FBER measurement. Default is Zero . | RW |
| 1 | Clear FBER Counters – Writing a 1b to this bit clears the FBER counters. This bit always return Zero when read | RW |
| 3:2 | Granularity of per-Lane Error reported – 00b count all bit errors corrected by FEC in valid Flits, 01b count all even bit errors in each UI corrected by FEC in valid Flits, 10b count all odd bit errors in each UI corrected by FEC in valid Flits, 11b count only mismatches in the Control SKP OS as a single correctable bit error FBER measurement results are undefined if this field contains 01b or 10b and the Link is not operating in PAM4. Default is Zero . | RW |
| 4 | Report Longest Burst vs First Burst – This bit is now deprecated. Behavior is undefined if this bit is Set. Default is Zero . | RW |

7.7.8.7 FBER Measurement Status 1 Register (Offset 14h) §

FBER Measurement Status 1 Register through FBER Measurement Status 10 Register contain a collection of per-Link and per-Lane counters:

- Flit Counter – per-Link
- Invalid Flit Counter – per-Link
- Correctable Error Counters – per-Lane (Adding these together produces a per-Link counter)

A write of 1b to either the Clear FBER Counters or the FBER Measurement Enable bit of the FBER Measurement Control Register resets the value of all of these counters to their default values. All of these counters saturate (i.e., when they reach their maximum value, they do not roll over).

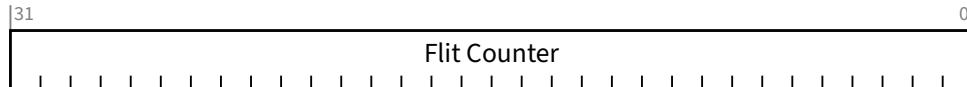


Figure 7-110 FBER Measurement Status 1 Register

Table 7-98 FBER Measurement Status 1 Register

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:0 | <p>Flit Counter – meaningful when FBER Measurement Enable is Set Increments by 1 for every Flit received. Default is Zero.</p> | ROS |

7.7.8.8 FBER Measurement Status 2 Register (Offset 18h)

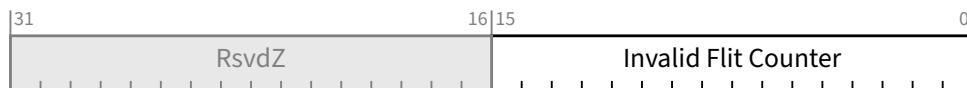


Figure 7-111 FBER Measurement Status 2 Register

Table 7-99 FBER Measurement Status 2 Register

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | <p>Invalid Flit Counter – when FBER Measurement Enable is Set: Increments by 1 for every invalid Flit received. Otherwise, behavior is undefined. Default is 0000h.</p> | ROS |

7.7.8.9 FBER Measurement Status 3 Register (Offset 1Ch) §

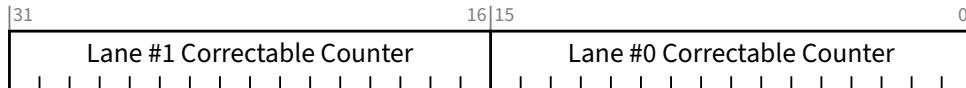


Figure 7-112 FBER Measurement Status 3 Register §

Table 7-100 FBER Measurement Status 3 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | <p>Lane #0 Correctable Counter – counts Per-Lane Correctable Bit Errors or SKP Parity Mismatches.</p> <p>If FBER Measurement Enable is Set: this 16-bit counter that counts the number of FEC-correctable bit errors per Flit (up to 24) or the number of SKP OS Parity mismatch in a Port, as per the value in <u>Granularity of per-Lane Error Reported</u> bit in the FBER Measurement Control Register .</p> <p>This counter does not roll-over.</p> <p>Default is Zero .</p> | RO |
| 31:16 | <p>Lane #1 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter .</p> | RO |

7.7.8.10 FBER Measurement Status 4 Register (Offset 20h) §

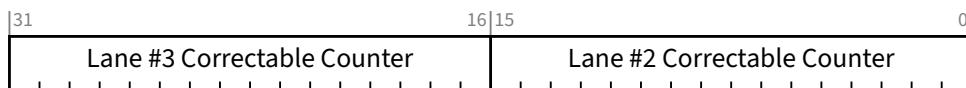


Figure 7-113 FBER Measurement Status 4 Register §

Table 7-101 FBER Measurement Status 4 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | <p>Lane #2 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter .</p> | RO |
| 31:16 | <p>Lane #3 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter .</p> | RO |

7.7.8.11 FBER Measurement Status 5 Register (Offset 24h) §

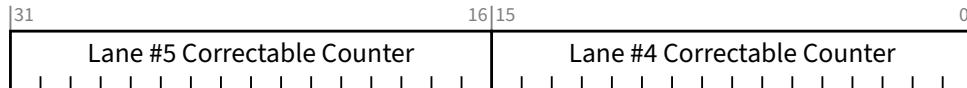


Figure 7-114 FBER Measurement Status 5 Register §

Table 7-102 FBER Measurement Status 5 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | Lane #4 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |
| 31:16 | Lane #5 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |

7.7.8.12 FBER Measurement Status 6 Register (Offset 28h) §

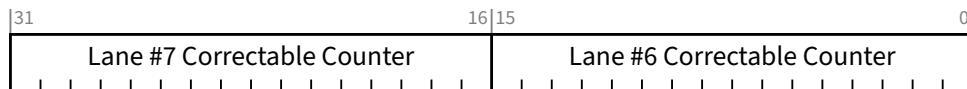


Figure 7-115 FBER Measurement Status 6 Register §

Table 7-103 FBER Measurement Status 6 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | Lane #6 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |
| 31:16 | Lane #7 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |

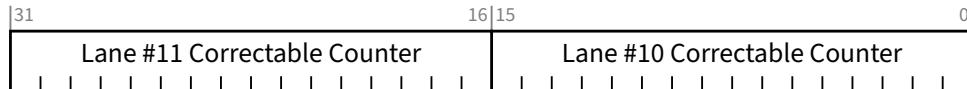
7.7.8.13 FBER Measurement Status 7 Register (Offset 2Ch) §



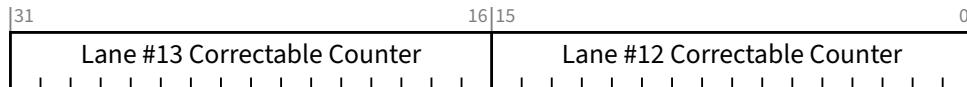
Figure 7-116 FBER Measurement Status 7 Register §

Table 7-104 FBER Measurement Status 7 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | Lane #8 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |
| 31:16 | Lane #9 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |

7.7.8.14 FBER Measurement Status 8 Register (Offset 30h) §*Figure 7-117 FBER Measurement Status 8 Register §**Table 7-105 FBER Measurement Status 8 Register §*

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Lane #10 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |
| 31:16 | Lane #11 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |

7.7.8.15 FBER Measurement Status 9 Register (Offset 34h) §*Figure 7-118 FBER Measurement Status 9 Register §**Table 7-106 FBER Measurement Status 9 Register §*

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Lane #12 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |
| 31:16 | Lane #13 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |

7.7.8.16 FBER Measurement Status 10 Register (Offset 38h) §

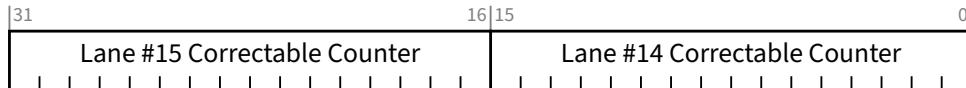


Figure 7-119 FBER Measurement Status 10 Register §

Table 7-107 FBER Measurement Status 10 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Lane #14 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |
| 31:16 | Lane #15 Correctable Counter – Behavior is identical to Lane #0 Correctable Counter . | RO |

7.7.9 Device 3 Extended Capability Structure §

The Device 3 Extended Capability structure must be implemented in any Function or RCRB that implements any mechanism that requires the registers in this Extended Capability. It is permitted for this Extended Capability to be implemented in Functions or RCRBs that do not require any of the registers in this Extended Capability.

§ Figure 7-120 details allocation of the register bits in the Device 3 Extended Capability structure.

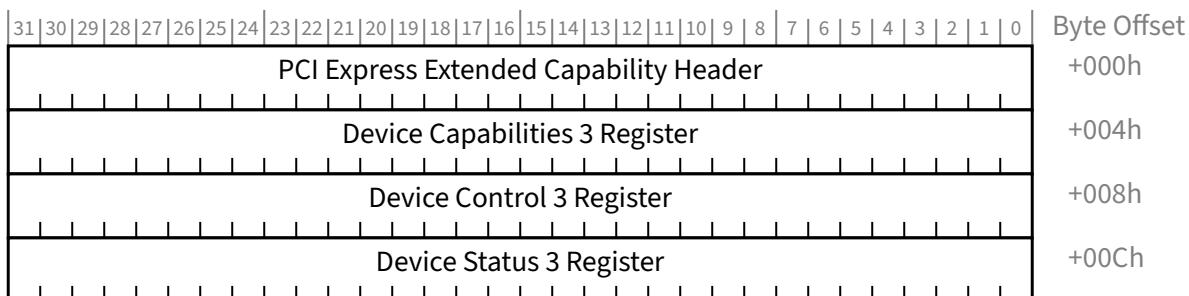


Figure 7-120 Device 3 Extended Capability Structure §

7.7.9.1 Device 3 Extended Capability Header (Offset 00h) §

§ Figure 7-121 details allocation of the register fields in the Device 3 Extended Capability Header ; § Table 7-108 provides the respective bit definitions.

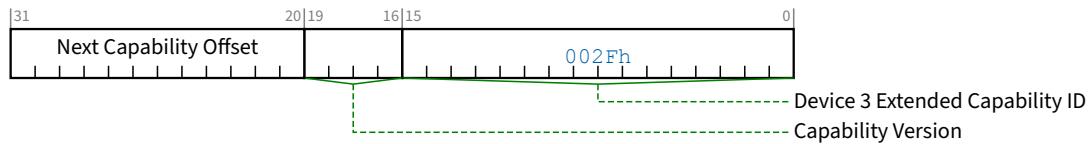


Table 7-108 Device 3 Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Device 3 Extended Capability ID - Indicates the Device 3 Extended Capability structure. This field must return a Capability ID of 002Fh indicating that this is a Device 3 Extended Capability structure. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

7.7.9.2 Device Capabilities 3 Register (Offset 04h) §

§ Figure 7-122 details the allocation of register bits of the Device Capability 3 register; § Table 7-109 provides the respective bit definitions.

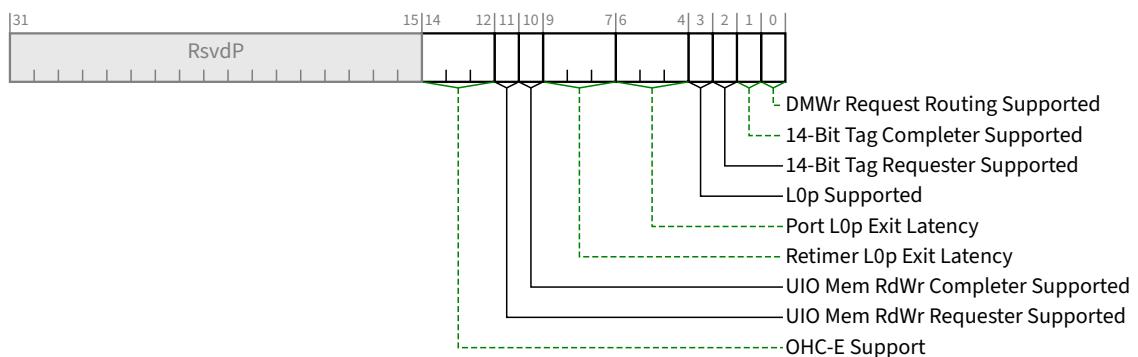


Table 7-109 Device Capabilities 3 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | DMWr Request Routing Supported - Applicable only to Switch Upstream Ports, Switch Downstream Ports, and Root Ports; must be 0b for other Function types. This bit must be Set if the Port supports this optional capability. See § Section 6.32 for additional details. | HwInit |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|-----------------------------|----------------|-------------|------------------------|-------------|------------------------|-------------|------------------------|-------------|-------------------------|-------------|--------------------------|-------------|-------------|-------------|-----------------|-----------------------------|
| 1 | <p>14-Bit Tag Completer Supported – If this bit is Set, the Function supports 14-Bit Tag Completer capability; otherwise, the Function does not. See § Section 2.2.6.2 for additional details.</p> <p>This bit MUST@FLIT be Set.</p> <p>For VFs, this bit value must be identical to the associated PF's bit value.</p> | HwInit | | | | | | | | | | | | | | | | |
| 2 | <p>14-Bit Tag Requester Supported – If this bit is Set, the Function supports 14-Bit Tag Requester ↓↓capability;↓↑capability for non-UIO Requests;↑ otherwise, the Function does not. ↓↑This bit is not applicable to a Requester when generating UIO Requests.↑</p> <p>This bit must not be Set if the 14-Bit Tag Completer Supported bit is Clear.</p> <p>If the Function is an RCiEP, this bit must be Clear if the RC does not support 14-Bit Tag Completer capability for Requests coming from this RCiEP.</p> <p>For VFs, this bit value must equal the VF 14-Bit Tag Requester Supported bit value in the SR-IOV Capabilities Register . See § Section 9.4.3.2.3 for additional details.</p> <p>Note that 14-Bit Tag field generation must be enabled by the 14-Bit Tag Requester Enable bit in the Device Control 3 register of the Requester Function before ↓↓non-UIO Requests with↑ 14-Bit Tags can be ↓↓generated by the Requester.↓↑generated.↑ See § Section 2.2.6.2 for additional details.</p> | Errata: Base 6.3 B807△◀▷ | | | | | | | | | | | | | | | | |
| 3 | <p>L0p Supported – If Set, the Port supports L0p . This bit must be ↓↓clear↓ ↑↑Clear if Flit Mode Supported is Clear.</p> <p>All Functions associated with an Upstream Port must return the same value of this bit.</p> <p>↓↑This bit is meaningful only in Downstream Port Functions and in Function 0 of Upstream Ports.↑</p> | Errata: Base 6.3 B823△◀▷ | | | | | | | | | | | | | | | | |
| 6:4 | <p>Port L0p Exit Latency – indicates this Port's L0p Exit Latency. The value reported indicates the length of time this Port requires to complete widening a link using L0p. If L0p Supported is clear, this field must contain 000b.</p> <p>All Functions associated with an Upstream Port must return the same value of this field.</p> <p>↓↑This bit is meaningful only in Downstream Port Functions and in Function 0 of Upstream Ports.↑</p> <p>Local L0p Exit Latency is computed as the maximum of Port L0p Exit Latency and Retimer L0p Exit Latency . Local L0p Exit Latency is transmitted in the L0p Exit Latency field of the Data Link Feature DLLP. The effective L0p Exit Latency of a Link is computed as the maximum of Local L0p Exit Latency and Remote L0p Exit Latency .</p> <p>Defined encodings are:</p> <table> <tbody> <tr> <td>000b</td> <td>Less than 1 μs</td> </tr> <tr> <td>001b</td> <td>1 μs to less than 2 μs</td> </tr> <tr> <td>010b</td> <td>2 μs to less than 4 μs</td> </tr> <tr> <td>011b</td> <td>4 μs to less than 8 μs</td> </tr> <tr> <td>100b</td> <td>8 μs to less than 16 μs</td> </tr> <tr> <td>101b</td> <td>16 μs to less than 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> </tbody> </table> | 000b | Less than 1 μs | 001b | 1 μs to less than 2 μs | 010b | 2 μs to less than 4 μs | 011b | 4 μs to less than 8 μs | 100b | 8 μs to less than 16 μs | 101b | 16 μs to less than 32 μs | 110b | 32 μs-64 μs | 111b | More than 64 μs | Errata: Base 6.3 B823△◀▷ |
| 000b | Less than 1 μs | | | | | | | | | | | | | | | | | |
| 001b | 1 μs to less than 2 μs | | | | | | | | | | | | | | | | | |
| 010b | 2 μs to less than 4 μs | | | | | | | | | | | | | | | | | |
| 011b | 4 μs to less than 8 μs | | | | | | | | | | | | | | | | | |
| 100b | 8 μs to less than 16 μs | | | | | | | | | | | | | | | | | |
| 101b | 16 μs to less than 32 μs | | | | | | | | | | | | | | | | | |
| 110b | 32 μs-64 μs | | | | | | | | | | | | | | | | | |
| 111b | More than 64 μs | | | | | | | | | | | | | | | | | |
| 9:7 | <p>Retimer L0p Exit Latency – indicates this worst case L0p Exit Latency for retimers “associated” with this Port. The value reported indicates the length of time a Retimer requires to complete widening a link using L0p. If L0p Supported is clear, this field must contain 000b.</p> | HwInit | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------------------------|
| | <p>All Functions associated with an Upstream Port must return the same value of this field.</p> <p>↑↑This bit is meaningful only in Downstream Port Functions and in Function 0 of Upstream Ports.↑</p> <p>Local L0p Exit Latency is computed as the maximum of Port L0p Exit Latency and Retimer L0p Exit Latency . Local L0p Exit Latency is transmitted in the L0p Exit Latency field of the Data Link Feature DLLP. The effective L0p Exit Latency of a Link is computed as the maximum of Local L0p Exit Latency and Remote L0p Exit Latency .</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b Less than 1 µs 001b 1 µs to less than 2 µs 010b 2 µs to less than 4 µs 011b 4 µs to less than 8 µs 100b 8 µs to less than 16 µs 101b 16 µs to less than 32 µs 110b 32 µs-64 µs 111b More than 64 µs | <p>Errata: Base 6.3 B823△◀▷</p> |
| 10 | UIO Mem RdWr Completer Supported – When Set, indicates the Function supports UIO Memory Read and UIO Memory Write as a Completer. | HwInit |
| 11 | UIO Mem RdWr Requester Supported – When Set, indicates the Function supports UIO Memory Read and/or UIO Memory Write as a Requester. | HwInit |
| 14:12 | <p>OHC-E Support – Indicates the maximum number of OHC-E DWs supported by this Function as a receiver in Flit Mode. See § Section 2.2.11 for important details. Values are:</p> <ul style="list-style-type: none"> 000b OHC-E support is not indicated. 001b OHC-E1 supported as targeted completer. For switches, this encoding additionally indicates all OHC-Ex forwarding is supported. 010b OHC-E1 and OHC-E2 supported as targeted completer. For switches, this encoding additionally indicates all OHC-Ex forwarding is supported. 011b OHC-E1 , OHC-E2 and OHC-E4 supported as targeted completer. For switches, this encoding additionally indicates all OHC-Ex forwarding is supported. 100b For Switches, OHC-E not supported as targeted completer but all OHC-Ex forwarding is supported. Reserved for others. 101b-110b Reserved 111b OHC-E not supported as targeted completer. For switches, this encoding also indicates OHC-Ex forwarding is not supported. <p>000b encoding is present for backward compatibility purposes. It's strongly recommended that newer devices use a non-zero value of this field so SW can correctly enumerate the ↑↓OHC-↑↑OHC-E↑↑E↓ capability of the function.</p> <p>This field can be present in RP, RCiEP, Switch USP and Endpoint.</p> <p>For RPs, this field indicates support for OHC-E on TLPs for which the RP is the targeted completer i.e., on TLPs that are not forwarded to any peer RP or a RCiEP. Different Root Ports are permitted to report different values for this field.</p> <p>For Switches this field is present in the Switch USP function and reports the combined OHC-E capability of the USP and all Downstream Ports under that USP .</p> <p>This field is RsvdP if Flit Mode Supported is Clear.</p> | HwInit / RsvdP |

7.7.9.3 Device Control 3 Register (Offset 08h) §

§ Figure 7-123 details the allocation of register bits of the Device Control 3 register; § Table 7-110 provides the respective bit definitions.

IMPLEMENTATION NOTE: USE OF UIO REQUEST 256B BOUNDARY DISABLE §

UIO is intended to be suitable for routing its Requests directly to memory controllers. For memory architectures that support interleaving, it is intended that a single UIO Request not target multiple memory controllers. When Clear, the UIO Request 256B Boundary Disable bit prevents UIO Requests from crossing naturally aligned 256-byte address boundaries, supporting interleaving granularities of that size and integer multiples of it.

Flit Mode Link efficiency for 256-byte UIO Requests is relatively high, and it increases only a few percent when maximum-sized 4-KB Requests are used. However, for cases where using larger UIO Requests is desired in order to increase Link efficiency and/or lower TLP rates, software may Set the UIO Request 256B Boundary Disable bit to enable larger Requests. Software should only do this if it knows there is no requirement for this Requester to honor 256-byte boundaries.

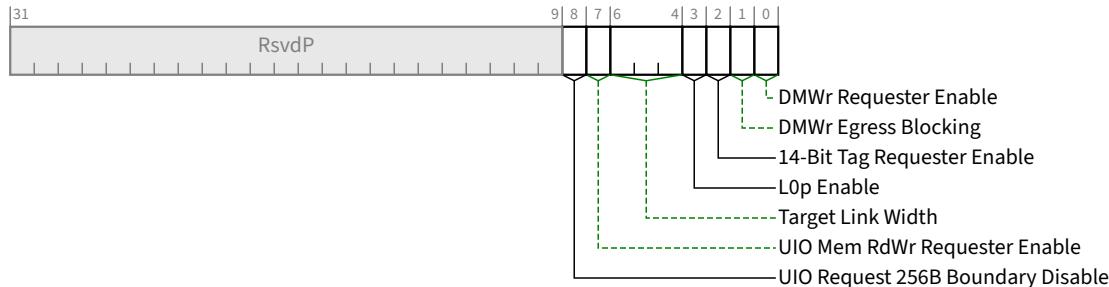


Figure 7-123 Device Control 3 Register §

Table 7-110 Device Control 3 Register §

| Bit Location ↑↓ | Register Description | Attributes |
|--------------------|--|--|
| 0 | <p>DMWr Requester Enable - Applicable only to Endpoints, Root Ports and RCRBs; must be hardwired to 0b for other Function types. The Function is allowed to initiate DMWr Requests only if this bit and the Bus Master Enable bit in the Command register are both Set.</p> <p>This bit is required to be RW if the Endpoint or Root Port is capable of initiating DMWr Requests, but otherwise is permitted to be hardwired to 0b.</p> <p>This bit does not serve as a capability bit. This bit is permitted to be RW even if no DMWr Requester capabilities are supported by the Endpoint or Root Port.</p> <p>Default value of this bit is 0b.</p> | RW ↑↓ / RO ↑ Errata: Base 6.3 B823△↔ ↑↓(see description)↑ |

Base 6.4 vs Base 6.3

| ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes | | | | | | | | | | | | | | |
|------------------------------------|--|--|---------|-------------|---------|-------------|---------|-------------|---------|-------------|----------|-------------|---|---------------|----------|------------|
| 1 | <p>DMWr Egress Blocking – Applicable and mandatory for Switch Upstream Ports, Switch Downstream Ports, and Root Ports that implement DMWr routing; otherwise must be hardwired to 0b.</p> <p>When this bit is Set, DMWr Requests that target going out this Egress Port must be blocked. See § Section 6.32 .</p> <p>Default value of this bit is 0b.</p> | RW / RO (see description) | | | | | | | | | | | | | | |
| 2 | <p>14-Bit Tag Requester Enable – This bit, in combination with the Extended Tag Field Enable bit and 10-Bit Tag Requester Enable bit, determines how many Tag field bits a Requester is permitted to use for non-UIO Requests. When ↓↑the 14-Bit Tag Requester Enable↓↑ this bit is Set, the Requester is permitted to use 14-Bit Tags. See § Section 2.2.6.2 for complete details.</p> <p>If software changes the value of this bit while the Function has outstanding Non-Posted Requests, the result is undefined.</p> <p>For VFs, this bit is not supported and is RsvdP . The value in the VF 14-Bit Tag Requester Enable bit in the associated PF's SR-IOV Control Register applies to all its VFs.</p> <p>Non-VF Functions that do not implement 14-Bit Tag Requester capability are permitted to hardwire this bit to 0b.</p> <p>Default value of this bit is 0b.</p> | RW / RO (see description) VF RsvdP | | | | | | | | | | | | | | |
| 3 | <p>L0p Enable – Determines behavior of this Port when sending or responding to Link Management DLLPs of type L0p DLLP .</p> <p>This bit has no effect on Link Management DLLPs where the Link Mgmt Type field is other than L0p DLLP .</p> <p>This bit ↓↑has no effect if↓↑is meaningful only in Downstream Port Functions and in Function 0 of Upstream Ports, and only when L0p Supported is Set and↓↑Hardware Autonomous Width Disable is ↓↑1b.↓↑Clear. In other Functions, this bit has no effect, and it is permitted to hardwire the bit to 0b.↑</p> <p>Default is 1b.</p> | RW ↓↑/ RO↑ ↑↑(see description)↑ | | | | | | | | | | | | | | |
| 6:4 | <p>Target Link Width – writes to this field initiate a directed L0p Link Width change to the indicated width. Encodings are:</p> <table style="margin-left: 20px;"> <tr><td>000b</td><td>x1 Link</td></tr> <tr><td>001b</td><td>x2 Link</td></tr> <tr><td>010b</td><td>x4 Link</td></tr> <tr><td>011b</td><td>x8 Link</td></tr> <tr><td>100b</td><td>x16 Link</td></tr> <tr><td>111b</td><td>Dynamic – L0p Link Width is determined by the Ports with no architected software intervention</td></tr> <tr><td>Others</td><td>Reserved</td></tr> </table> <p>This field has no effect on subsequent autonomous Link Width ↓↑changes. This field has no effect↓↑changes, or↑ on subsequent Link Width changes due to link reliability.</p> <p>This field does not represent maximum Link Width support.</p> | 000b | x1 Link | 001b | x2 Link | 010b | x4 Link | 011b | x8 Link | 100b | x16 Link | 111b | Dynamic – L0p Link Width is determined by the Ports with no architected software intervention | Others | Reserved | RW / RsvdP |
| 000b | x1 Link | | | | | | | | | | | | | | | |
| 001b | x2 Link | | | | | | | | | | | | | | | |
| 010b | x4 Link | | | | | | | | | | | | | | | |
| 011b | x8 Link | | | | | | | | | | | | | | | |
| 100b | x16 Link | | | | | | | | | | | | | | | |
| 111b | Dynamic – L0p Link Width is determined by the Ports with no architected software intervention | | | | | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| ↓↑Bit Location↓ ↓↑Bit Location↑ | Register Description | Attributes |
|------------------------------------|---|------------------------------|
| | <p>This field is RsvdP if Flit Mode Supported is Clear. This field is permitted in RCRBs.</p> <p>This field has no effect if L0p Enable is Clear.</p> <p>This field ↓↓has no effect if↓↓is meaningful only in Downstream Port Functions and in Function 0 of Upstream Ports, and only when L0p Enable is Set and↓↓ Hardware Autonomous Width Disable is ↓↓1b.↓↓Clear. In other Functions, this field has no effect, and it is permitted to hardwire the field to 111b.↑↑</p> <p>Behavior is undefined if this field is set to a reserved encoding or to a width that is greater than the Link width on the most recent entry to L0 .</p> <p>Default is 111b.</p> | |
| 7 | <p>UIO Mem RdWr Requester Enable – The Function is permitted to initiate UIO Memory Read and UIO Memory Write only if this bit and the Bus Master Enable bit in the Command Register are both Set.</p> <p>This bit is required to be RW if UIO Mem RdWr Requester Supported is Set, but otherwise is permitted to be hardwired to 0b.</p> <p>Default value of this bit is 0b.</p> | RW / RO (see description) |
| 8 | <p>UIO Request 256B Boundary Disable – When Clear, a UIO Request from this Function must not specify an Address/Length combination that causes a Memory Space access to cross a naturally aligned 256-byte boundary. When Set, a Request from this Function may cross a naturally aligned 256B boundary. The setting of this bit has no impact on the separate requirement that all Memory Requests must not specify an Address/Length combination that causes a Memory Space access to cross a naturally aligned 4-KB boundary (see § Section 2.2.7).</p> <p>Mechanisms outside the scope of this specification may enable more advanced boundary policies, such as using larger or smaller boundaries than 256B, or boundaries associated with specific address ranges. However, such policies must never violate the boundary requirements stated in this description. See the Implementation Note: Use of UIO Request 256B Boundary Disable.</p> <p>This bit is permitted to be hardwired to 0b if this Function's UIO Mem RdWr Requester Supported bit is Clear.</p> <p>Default value of this bit is 0b.</p> | RW / RO (see description) |

7.7.9.4 Device Status 3 Register (Offset 0Ch) §

§ Figure 7-124 details allocation of the register fields in the Device Status 3 Register ; § Table 7-111 provides the respective bit definitions.

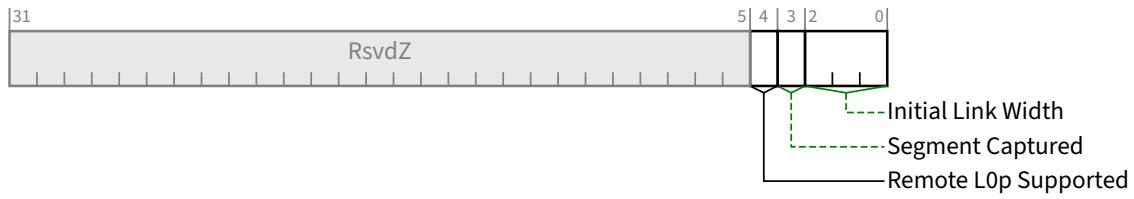


Figure 7-124 Device Status 3 Register §

Table 7-111 Device Status 3 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 2:0 | <p>Initial Link Width – This field contains the Link Width determined during initial link training. Encodings are:</p> <ul style="list-style-type: none"> 000b x1 Link 001b x2 Link 010b x4 Link 011b x8 Link 100b x16 Link Others Reserved <p>Default is determined during initial link training. Note that the current Link Width is visible in the Negotiated Link Width field.</p> | RO |
| 3 | <p>Segment Captured – This bit indicates if the Function has captured a valid Segment value from a Configuration Write Request as described in § Section 2.2.6.2 . When the Destination Segment field is captured from a Configuration Write Request in FM this bit must be set to the value of the DSV bit received with the Request. This bit must be cleared when a Configuration Write Request is received in NFM.</p> <p>Note that this bit will be set when every Link on the path from the Function to the RC is in FM. This bit will be clear if any Link between the Function and the RC is in NFM. Functions should only initiate Route by ID Message Requests targeting Hierarchies other than their own when this bit is Set.</p> <p>This bit is permitted to be hardwired to 0b in devices that don't support FM.</p> <p>FM Requesters and Completers within an RC capture their Segment value in an implementation specific way and must then Set this bit.</p> <p>The value in a Switch Downstream Port must be identical to the value in the associated Switch Upstream Port.</p> <p>Default is Zero in Functions that capture their Segment value.</p> | RO |
| 4 | <p>Remote L0p Supported – This bit indicates that the remote end of the Link supports L0p.</p> <p>Default is zero.</p> | RO |

7.7.10 Lane Margining at the Receiver Extended Capability §

The Lane Margining at the Receiver Extended Capability structure must be implemented in:

- A Function associated with a Downstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s or higher.
- A Function of a Single-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s or higher.
- Function 0 (and only Function 0) of a Multi-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s or higher.

§ Figure 7-125 shows the layout of the Margining Extended Capability. This capability contains a pair of per-Port registers followed by a set of per-Lane registers.

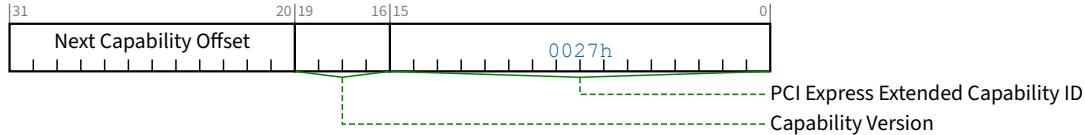
The number of per-Lane entries is determined by the Maximum Link Width (see § Section 7.5.3.6 or § Section 7.9.9.2). Up to 32 entries are permitted regardless of the Maximum Link Width. The value of entries beyond the Maximum Link Width is undefined.

Each per-Lane entry contains the values for that Lane. Lane numbering uses the ~~↓↑default Lane number and is thus invariant to Link width and Lane reversal negotiation that occurs during Link training.~~ ↓↑Physical Lane Number.

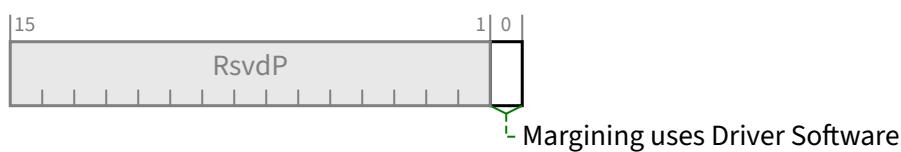
Errata: Base 6.3
B825△◀

Base 6.4 vs Base 6.3

| Byte Offset | |
|-------------|--|
| +000h | PCI Express Extended Capability Header |
| +004h | Margining Port Status Register |
| +008h | Margining Port Capabilities Register |
| +00Ch | Margining Lane Status: Lane 0 |
| +010h | Margining Lane Control: Lane 0 |
| +014h | Margining Lane Status: Lane 1 (Optional) |
| +018h | Margining Lane Control: Lane 1 (Optional) |
| +01Ch | Margining Lane Status: Lane 2 (Optional) |
| +020h | Margining Lane Control: Lane 2 (Optional) |
| +024h | Margining Lane Status: Lane 3 (Optional) |
| +028h | Margining Lane Control: Lane 3 (Optional) |
| +02Ch | Margining Lane Status: Lane 4 (Optional) |
| +030h | Margining Lane Control: Lane 4 (Optional) |
| +034h | Margining Lane Status: Lane 5 (Optional) |
| +038h | Margining Lane Control: Lane 5 (Optional) |
| +03Ch | Margining Lane Status: Lane 6 (Optional) |
| +040h | Margining Lane Control: Lane 6 (Optional) |
| +044h | Margining Lane Status: Lane 7 (Optional) |
| +048h | Margining Lane Control: Lane 7 (Optional) |
| +04Ch | Margining Lane Status: Lane 8 (Optional) |
| +050h | Margining Lane Control: Lane 8 (Optional) |
| +054h | Margining Lane Status: Lane 9 (Optional) |
| +058h | Margining Lane Control: Lane 9 (Optional) |
| +05Ch | Margining Lane Status: Lane 10 (Optional) |
| +060h | Margining Lane Control: Lane 10 (Optional) |
| +064h | Margining Lane Status: Lane 11 (Optional) |
| +068h | Margining Lane Control: Lane 11 (Optional) |
| +06Ch | Margining Lane Status: Lane 12 (Optional) |
| +070h | Margining Lane Control: Lane 12 (Optional) |
| +074h | Margining Lane Status: Lane 13 (Optional) |
| +078h | Margining Lane Control: Lane 13 (Optional) |
| +07Ch | Margining Lane Status: Lane 14 (Optional) |
| +080h | Margining Lane Control: Lane 14 (Optional) |
| +084h | Margining Lane Status: Lane 15 (Optional) |
| | Margining Lane Control: Lane 15 (Optional) |
| | Margining Lane Status: Lane 16 (Optional) |
| | Margining Lane Control: Lane 16 (Optional) |
| | Margining Lane Status: Lane 17 (Optional) |
| | Margining Lane Control: Lane 17 (Optional) |
| | Margining Lane Status: Lane 18 (Optional) |
| | Margining Lane Control: Lane 18 (Optional) |
| | Margining Lane Status: Lane 19 (Optional) |
| | Margining Lane Control: Lane 19 (Optional) |
| | Margining Lane Status: Lane 20 (Optional) |
| | Margining Lane Control: Lane 20 (Optional) |
| | Margining Lane Status: Lane 21 (Optional) |
| | Margining Lane Control: Lane 21 (Optional) |
| | Margining Lane Status: Lane 22 (Optional) |
| | Margining Lane Control: Lane 22 (Optional) |
| | Margining Lane Status: Lane 23 (Optional) |
| | Margining Lane Control: Lane 23 (Optional) |
| | Margining Lane Status: Lane 24 (Optional) |
| | Margining Lane Control: Lane 24 (Optional) |
| | Margining Lane Status: Lane 25 (Optional) |
| | Margining Lane Control: Lane 25 (Optional) |
| | Margining Lane Status: Lane 26 (Optional) |
| | Margining Lane Control: Lane 26 (Optional) |
| | Margining Lane Status: Lane 27 (Optional) |
| | Margining Lane Control: Lane 27 (Optional) |
| | Margining Lane Status: Lane 28 (Optional) |
| | Margining Lane Control: Lane 28 (Optional) |
| | Margining Lane Status: Lane 29 (Optional) |
| | Margining Lane Control: Lane 29 (Optional) |
| | Margining Lane Status: Lane 30 (Optional) |
| | Margining Lane Control: Lane 30 (Optional) |
| | Margining Lane Status: Lane 31 (Optional) |
| | Margining Lane Control: Lane 31 (Optional) |

*Figure 7-125 Lane Margining at the Receiver Extended Capability***7.7.10.1 Lane Margining at the Receiver Extended Capability Header (Offset 00h)***Figure 7-126 Lane Margining at the Receiver Extended Capability Header**Table 7-112 Lane Margining at the Receiver Extended Capability Header*

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Lane Margining at the Receiver Extended Capability is 0027h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.7.10.2 Margining Port Capabilities Register (Offset 04h)*Figure 7-127 Margining Port Capabilities Register**Table 7-113 Margining Port Capabilities Register*

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Margining uses Driver Software - If Set, indicates that Margining is partially implemented using Device Driver software. Margining Software Ready indicates when this software is initialized. If Clear, Margining does not require device driver software. In this case the value read from Margining Software Ready is undefined. | HwInit |

7.7.10.3 Margining Port Status Register (Offset 06h) §

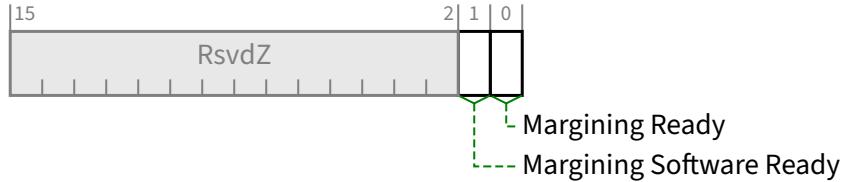


Figure 7-128 Margining Port Status Register §

Table 7-114 Margining Port Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Margining Ready - Indicates when the Margining feature is ready to accept margining commands. Behavior is undefined if this bit is Clear and, for any Lane, any of the Receiver Number, Margin Type, Usage Model, or Margin Payload fields are written (see § Section 7.7.10.4).</p> <p>If Margining uses Driver Software is Set, Margining Ready must be Set no later than 100 ms after the later of Margining Software Ready becoming Set or the link training to 16.0 GT/s or higher.</p> <p>If Margining uses Driver Software is Clear, Margining Ready must be Set no later than 100 ms after the Link trains to 16.0 GT/s or higher.</p> <p>Default value is implementation specific.</p> | RO |
| 1 | <p>Margining Software Ready - When Margining uses Driver Software is Set, then this bit, when Set, indicates that the required software has performed the required initialization.</p> <p>The value of this bit is undefined if Margining uses Driver Software is Clear. The default value of this bit is implementation specific.</p> | RO |

7.7.10.4 Margining Lane Control Register (Offset 08h) §

The Margining Lane Control Register consists of control fields required for per-Lane margining.

The number of entries in this register are sized by Maximum Link Width (see § Section 7.5.3.6).

See § Section 4.2.8.2 for details of this register.

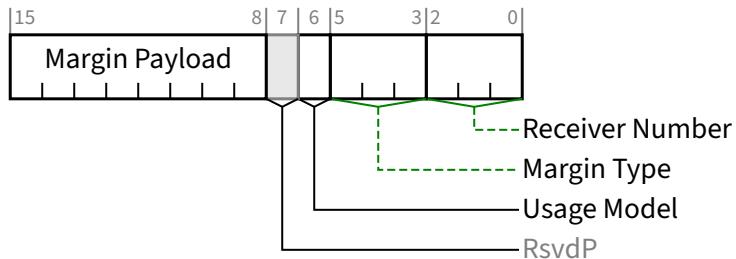


Figure 7-129 Lane N: Margining Control Register Entry §

Table 7-115 Lane N: Margining Control Register Entry §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------|
| 2:0 | <p>Receiver Number - See § Section 4.2.18.1 for details. The default value is 000b. This field must be reset to the default value if the Port goes to DL_Down status.</p> | RW (see description) |
| 5:3 | <p>Margin Type - See § Section 4.2.18.1 for details. The default value is 111b. This field must be reset to the default value if the Port goes to DL_Down status.</p> | RW (see description) |
| 6 | <p>Usage Model - See § Section 4.2.18.1 for details. The default value is 0b. This field must be reset to the default value if the Port goes to DL_Down status.</p> | RW (see description) |
| 15:8 | <p>Margin Payload - See § Section 4.2.18.1 for details. This field's value is used in conjunction with the Margin Type field, as described in § Section 4.2.18.1. The default value is 9Ch. This field must be reset to the default value if the Port goes to DL_Down status.</p> | RW (see description) |

7.7.10.5 Margining Lane Status Register (Offset 0Ah) §

The Margining Lane Status register consists of status fields required for per-Lane margining. The number of entries in this register are sized by Maximum Link Width (see § Section 7.5.3.6). See § Section 4.2.8.2 for details of this register.

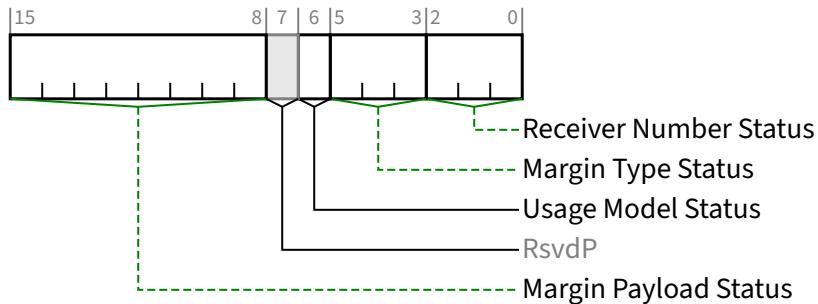


Figure 7-130 Lane N: Margin Lane Status Register Entry §

Table 7-116 Lane N: Margin Lane Status Register Entry §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------|
| 2:0 | Receiver Number Status - See § Section 4.2.18.1 for details. The default value is 000b. For Downstream Ports, this field must be reset to the default value if the Port goes to DL_Down status. | RO (see description) |
| 5:3 | Margin Type Status - See § Section 4.2.18.1 for details. The default value is 000b. This field must be reset to the default value if the Port goes to DL_Down status. | RO (see description) |
| 6 | Usage Model Status - See § Section 4.2.18.1 for details. The default value is 0b. This field must be reset to the default value if the Port goes to DL_Down status. | RO (see description) |
| 15:8 | Margin Payload Status - See § Section 4.2.18.1 for details. This field is only meaningful, when the Margin Type is a defined encoding other than 'No Command'. The default value is 00h. This field must be reset to the default value if the Port goes to DL_Down status. | RO (see description) |

7.7.11 ACS Extended Capability §

The ACS Extended Capability is an optional capability that provides enhanced access controls (see § Section 6.12). This capability may be implemented by a Root Port, a Switch Downstream Port, or a Multi-Function Device Function. It is never applicable to a PCI Express to PCI Bridge or Root Complex Event Collector. It is not applicable to a Switch Upstream Port unless that Switch Upstream Port is a Function in a Multi-Function Device .

If an **SR-IOV Capable Device** **SR-IOV Device or an SIOV Device** other than one in a Root Complex implements internal peer-to-peer transactions, ACS is required, and ACS P2P Egress Control must be supported.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

Implementation of ACS in RCiEPs is permitted but not required. It is explicitly permitted that within a single Root Complex, some RCiEPs implement ACS and some do not. It is strongly recommended that Root Complex implementations ensure that all accesses originating from RCiEPs (PFs and VFs) without ACS capability are first

subjected to processing by a Translation Agent (TA) in the Root Complex before further decoding and processing. The details are outside the scope of this specification.

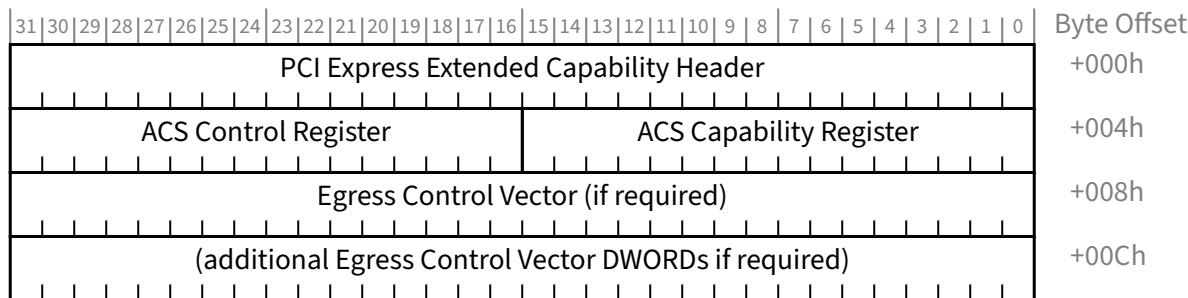


Figure 7-131 ACS Extended Capability §

7.7.11.1 ACS Extended Capability Header (Offset 00h) §

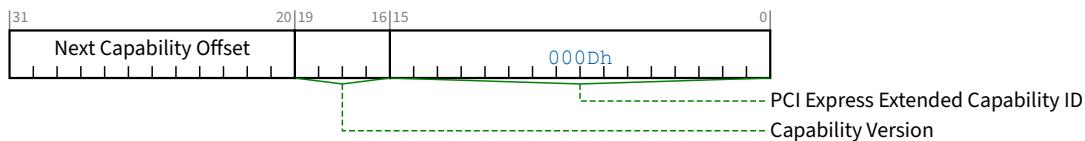


Figure 7-132 ACS Extended Capability Header §

Table 7-117 ACS Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the ACS Extended Capability is 000Dh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

7.7.11.2 ACS Capability Register (Offset 04h) §

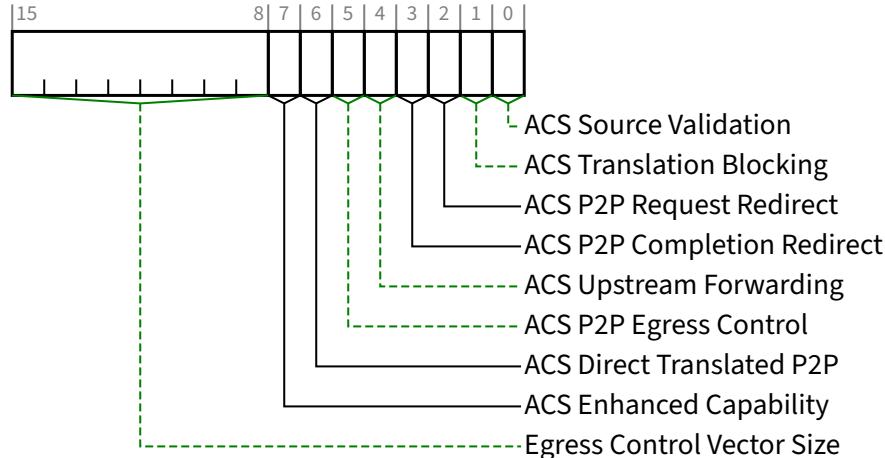


Figure 7-133 ACS Capability Register §

Table 7-118 ACS Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|--|---|
| 0 | ACS Source Validation - Required for Root Ports and Switch Downstream Ports; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Source Validation . | RO |
| 1 | ACS Translation Blocking - Required for Root Ports and Switch Downstream Ports; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Translation Blocking . | RO |
| 2 | ACS P2P Request Redirect - Required for Root Ports that support peer-to-peer traffic with other Root Ports; required for Switch Downstream Ports; required for Multi-Function Device Functions that support peer-to-peer traffic with other Functions; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS P2P Request Redirect . | RO |
| 3 | ACS P2P Completion Redirect - Required for all Functions that support ACS P2P Request Redirect ; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS P2P Completion Redirect . | RO |
| 4 | ACS Upstream Forwarding - Required for Root Ports if the RC supports Redirected Request Validation; required for Switch Downstream Ports; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Upstream Forwarding . | RO |
| 5 | ACS P2P Egress Control - Except as stated below, optional for Root Ports, Switch Downstream Ports, and Multi-Function Device Functions; otherwise this bit must be hardwired to Zero. If Set, indicates that the component implements ACS P2P Egress Control. For an ↑↓SR-IOV Device ↑↓SR-IOV Device or an SIOV Device not in a Root Complex, this bit is required to be Set for Functions if peer-to-peer transactions within the Device are supported. | RO <small>ECN: Base 6.3 SIOV△↔</small> |
| 6 | ACS Direct Translated P2P - Required for Root Ports that support Address Translation Services (ATS) and also support peer-to-peer traffic with other Root Ports; required for Switch Downstream Ports; required for Multi-Function Device Functions that support Address Translation Services (ATS) and also support | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | peer-to-peer traffic with other Functions; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Direct Translated P2P . | |
| 7 | <p>ACS Enhanced Capability - Required for Root Ports and Switch Downstream Ports that support the ACS Enhanced Capability mechanisms.</p> <p>If Set, indicates that the component supports all of the following mechanisms that are applicable:</p> <ul style="list-style-type: none"> ACS I/O Request Blocking ACS DSP Memory Target Access ACS USP Memory Target Access ACS Unclaimed Request Redirect | RO |
| 15:8 | <p>Egress Control Vector Size - Encodings 01h-FFh directly indicate the number of applicable bits in the Egress Control Vector ; the encoding 00h indicates 256 bits.</p> <p>If the ACS P2P Egress Control bit is 0b, the value of the size field is undefined, and the <u>Egress Control Vector Register</u> is not required to be present.</p> | HwInit |

7.7.11.3 ACS Control Register (Offset 06h) §

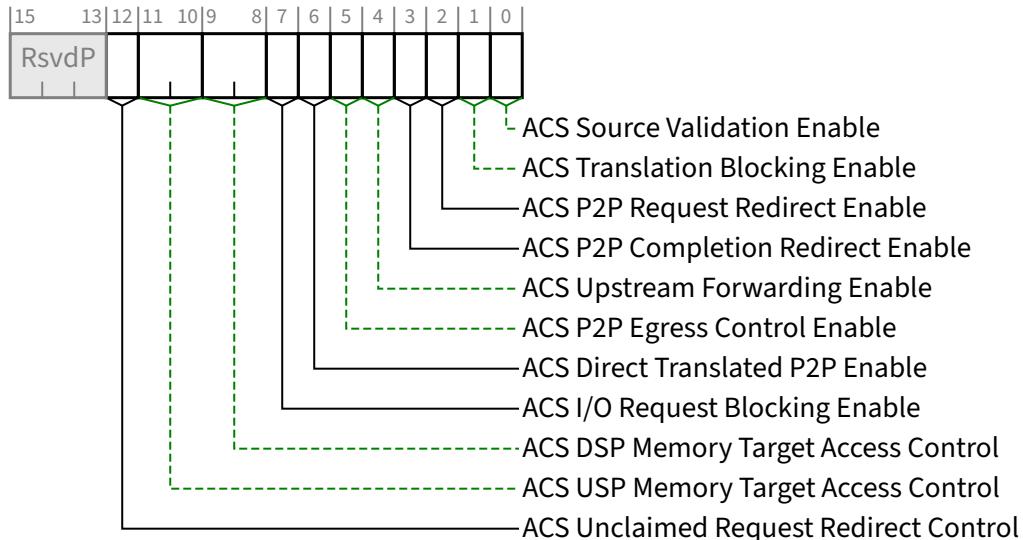


Figure 7-134 ACS Control Register §

Table 7-119 ACS Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>ACS Source Validation Enable - When Set, the component validates the Bus Number from the Requester ID of Upstream Requests against the secondary/subordinate Bus Numbers.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the <u>ACS Source Validation</u> functionality is not implemented.</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|--------------------------------|
| 1 | <p>ACS Translation Blocking Enable - When Set, the component blocks all Upstream Memory Requests whose Address Type (AT) field is not set to the default value.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS Translation Blocking functionality is not implemented.</p> | RW |
| 2 | <p>ACS P2P Request Redirect Enable - In conjunction with ACS P2P Egress Control and ACS Direct Translated P2P mechanisms, determines when the component redirects peer-to-peer Requests Upstream (see § Section 6.12.3). Note that with Downstream Ports, this bit only applies to Upstream Requests arriving at the Downstream Port, and whose normal <ins>non-ACSt</ins> routing targets a different Downstream Port.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Request Redirect functionality is not implemented.</p> | RW Errata: Base 6.3 B836△◀▶ |
| 3 | <p>ACS P2P Completion Redirect Enable - Determines when the component redirects peer-to-peer Completions Upstream; applicable only to Completions¹⁸⁴ whose Relaxed Ordering Attribute is clear.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Completion Redirect functionality is not implemented.</p> | RW |
| 4 | <p>ACS Upstream Forwarding Enable - When Set, the component forwards Upstream any Request or Completion TLPs it receives that were redirected Upstream by a component lower in the hierarchy. Note that this bit only applies to Upstream TLPs arriving at a Downstream Port, and whose normal routing targets <ins>would have targeted</ins> the same Downstream Port <ins>Port had</ins> this bit been Clear.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS Upstream Forwarding functionality is not implemented.</p> | RW Errata: Base 6.3 B836△◀▶ |
| 5 | <p>ACS P2P Egress Control Enable - In conjunction with the Egress Control Vector plus the ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms, determines when to allow, disallow, or redirect peer-to-peer Requests (see § Section 6.12.3).</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Egress Control functionality is not implemented.</p> | RW |
| 6 | <p>ACS Direct Translated P2P Enable - When Set, overrides the ACS P2P Request Redirect and ACS P2P Egress Control mechanisms with peer-to-peer Memory Requests whose Address Type (AT) field indicates a Translated address (see § Section 6.12.3).</p> <p>This bit is ignored if ACS Translation Blocking Enable is 1b.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS Direct Translated P2P functionality is not implemented.</p> | RW |
| 7 | <p>ACS I/O Request Blocking Enable - if Set, Upstream I/O Requests received by the Downstream Port must be handled as ACS Violations.</p> <p>This bit is required for Root Ports and Switch Downstream Ports if the ACS Enhanced Capability bit is Set; otherwise it must be RsvdP . The default value of this bit is 0b.</p> | RW / RsvdP |
| 9:8 | <p>ACS DSP Memory Target Access Control - This field controls how a Downstream Port handles Upstream Memory Requests attempting to access any Memory BAR Space on an applicable Root Port or Switch Downstream Port (including the Ingress Port). See § Section 6.12.1.1 .</p> <p>Defined Encodings are:</p> <ul style="list-style-type: none"> 00b Direct Request access enabled 01b Request blocking enabled 10b Request redirect enabled | RW / RsvdP |

¹⁸⁴. This includes Read Completions, AtomicOp Completions, and other Completions with or without Data.

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------|
| | <p>11b Reserved This field is required for Root Ports and Switch Downstream Ports if the ACS Enhanced Capability bit is Set and there is applicable Memory BAR Space to protect; otherwise it must be <u>RsvdP</u>. The default value of this field is 00b.</p> | |
| 11:10 | <p>ACS USP Memory Target Access Control - This field controls how a Switch Downstream Port handles Upstream Memory Requests attempting to access any Memory BAR Space on the Switch Upstream Port. See § Section 6.12.1.1 .</p> <p>Defined Encodings are:</p> <ul style="list-style-type: none"> 00b Direct Request access enabled 01b Request blocking enabled 10b Request redirect enabled 11b Reserved <p>This field is required for Switch Downstream Ports if the ACS Enhanced Capability bit is Set and there is applicable Memory BAR Space to protect; otherwise it must be <u>RsvdP</u>. The default value of this field is 00b.</p> | RW / <u>RsvdP</u> |
| 12 | <p>ACS Unclaimed Request Redirect Control - Controls how a Switch Downstream Port handles incoming Requests targeting Memory Space within the Memory aperture of the Switch Upstream Port that is not within a Memory aperture or Memory BAR Space of any Downstream Port within the Switch.</p> <p>When Set, the Switch must forward such Requests Upstream out of the Switch.</p> <p>When Clear, the Switch Downstream Port must handle such Requests as an Unsupported Request (UR).</p> <p>This bit is required for Switch Downstream Ports if the ACS Enhanced Capability bit is Set; otherwise it must be <u>RsvdP</u>. The default value of this bit is 0b.</p> | RW / <u>RsvdP</u> |

7.7.11.4 Egress Control Vector Register (Offset 08h) §

The Egress Control Vector is a read-write register that contains a bit-array. The number of bits in the register is specified by the Egress Control Vector Size field, and the register spans multiple DWORDs if required. If the ACS P2P Egress Control bit in the ACS Capability Register is 0b, the Egress Control Vector Size field is undefined and the Egress Control Vector Register is not required to be present.

For the general case of an Egress Control Vector spanning multiple DWORDs, the DWORD offset and bit number within that DWORD for a given arbitrary bit K are specified by the formulas¹⁸⁵:

$$\text{DWORD offset} = 08h + (K \text{ div } 32) \times 4$$

$$\text{DWORD bit\#} = K \text{ mod } 32$$

Equation 7-4 Egress Control Vector Access §

Bits in a DWORD beyond those specified by the Egress Control Vector Size field are RsvdP.

For Root Ports and Switch Downstream Ports, each bit in the bit-array always corresponds to a Port Number. Otherwise, for Functions¹⁸⁶ within a Multi-Function Device, each bit in the bit-array corresponds to one or more Function Numbers, or a Function Group Number. For example, access to Function 2 is controlled by bit number 2 in the bit-array. For both

¹⁸⁵. Div is an integer divide with truncation. Mod is the remainder from an integer divide.

¹⁸⁶. Including Switch Upstream Ports.

Port Number cases and Function Number cases, the bit corresponding to the Function that implements this Extended Capability structure must be hardwired to 0b.¹⁸⁷

If an ARI Device implements ACS Function Groups (ACS Function Groups Capability is Set), its Egress Control Vector Size is required to be a power-of-2 from 8 to 256, and all of its implemented Egress Control Vector bits must be RW. With ARI Devices, multiple Functions can be associated with a single bit, so for each Function, its associated bit determines how Requests from it targeting other Functions (if any) associated with the same bit are handled.

If ACS Function Groups are enabled in an ARI Device (ACS Function Groups Enable is Set), the first 8 Egress Control Vector bits in each Function are associated with Function Group Numbers instead of Function Numbers. In this case, access control is enforced between Function Groups instead of Functions, and any implemented Egress Control Vector bits beyond the first 8 are unused.

Independent of whether an ARI Device implements ACS Function Groups, its Egress Control Vector Size is not required to cover the entire Function Number range of all Functions implemented by the Device. If ACS Function Groups are not enabled, Function Numbers are mapped to implemented Egress Control Vector bits by taking the modulo of the Egress Control Vector Size, which is constrained to be a power-of-2.

With RCs, some Port Numbers may refer to internal Ports instead of Root Ports. For Root Ports in such RCs, each bit in the bit-array that corresponds to an internal Port must be hardwired to 0b.

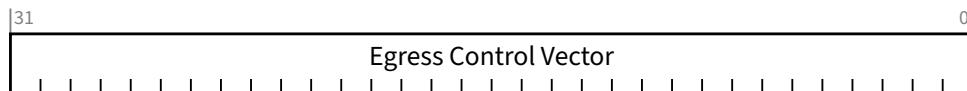


Figure 7-135 Egress Control Vector Register §

Table 7-120 Egress Control Vector Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:0 | <p>Egress Control Vector - An N-bit bit-array configured by software, where N is given by the value in the Egress Control Vector Size field. When a given bit is Set, peer-to-peer Requests targeting the associated Port, Function, or Function Group are blocked or redirected (if enabled) (see § Section 6.12.3).</p> <p>§ Figure 7-135 shows a single DWORD register. This register is always an integral number of DWORDs.</p> <p>Default value of each bit is 0b.</p> | RW |

The following examples illustrate how the vector might be configured:

- For an 8-Port Switch, each Port will have a separate vector indicating which Downstream Egress Ports it may forward Requests to.
Port 1 being not allowed to communicate with any other Downstream Ports would be configured as: 1111 1100b with bit 0 corresponding to the Upstream Port (hardwired to 0b) and bit 1 corresponding to the Ingress Port (hardwired to 0b).
Port 2 being allowed to communicate with Ports 3, 5, and 7 would be configured as: 0101 0010b.
- For a 4-Function device, each Function will have a separate vector that indicates which Function it may forward Requests to.
Function 0 being not allowed to communicate with any other Functions would be configured as: 1110b with bit 0 corresponding to Function 0 (hardwired to 0b).

187. For ARI Devices, the bit must be RW. See subsequent description.

Function 1 being allowed to communicate with Functions 2 and 3 would be configured as: 0001b with bit 1 corresponding to Function 1 (hardwired to 0b).

7.8 Common PCI and PCIe Capabilities §

This section contains a description of common PCI and PCIe capabilities that are individually optional in this but may be required by other PCISIG specifications.

7.8.1 Power Budgeting Extended Capability §

The Power Budgeting Extended Capability allows the system to allocate power to devices that are added to the system at runtime. Through this Capability, a device can report the power it consumes on a variety of power rails, in a variety of device power-management states, in a variety of operating conditions. The system can use this information to ensure that the system is capable of providing the proper power and cooling levels to the device. Failure to indicate proper device power consumption may risk device or system failure.

Implementation of the Power Budgeting Extended Capability is optional for PCI Express devices that are implemented either in a form-factor which does not require Hot-Plug support, or that are integrated on the system board. PCI Express form-factor specifications may require support for power budgeting. Power Budgeting reports device power consumption assuming the device is given appropriate permission (e.g., from Set_Slot_Power_Limit message) and that the external power connections for the device are operating.

The Power Budgeting Extended Capability is permitted to be present in PFs, but VFs must not implement it. If a PF contains the capability, it must report values that cover all associated \downarrow VFs \downarrow , \uparrow VFs or SDIs \uparrow .

§ Figure 7-136 details allocation of register fields in the Power Budgeting Extended Capability .

ECN: Base 6.3

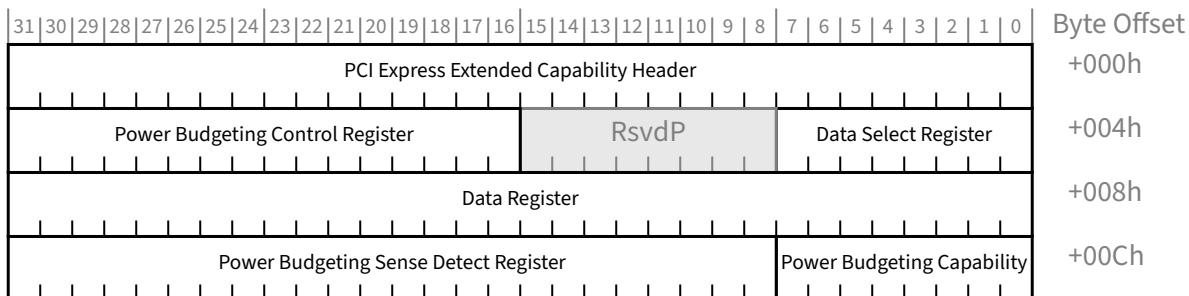


Figure 7-136 Power Budgeting Extended Capability §

7.8.1.1 Power Budgeting Extended Capability Header (Offset 00h) §

§ Figure 7-137 details allocation of register fields in the Power Budgeting Extended Capability Header; § Table 7-121 provides the respective bit definitions. Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Power Budgeting Extended Capability is 0004h .

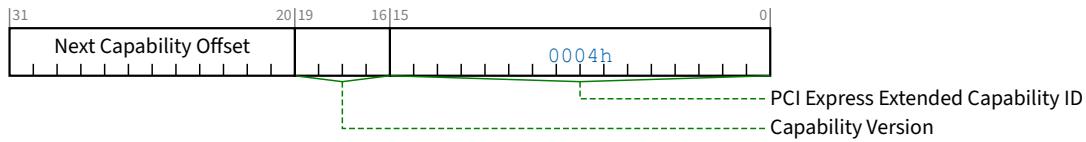


Figure 7-137 Power Budgeting Extended Capability Header §

Table 7-121 Power Budgeting Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Power Budgeting Extended Capability is 0004h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.8.1.2 Power Budgeting Data Select Register (Offset 04h) §

The Power Budgeting Data Select Register is an 8-bit read-write register that indexes the Power Budgeting Data reported through the Power Budgeting Data Register and selects the DWORD of Power Budgeting Data that is to appear in the Power Budgeting Data Register . Values for this register start at zero to select the first DWORD of Power Budgeting Data; subsequent DWORDs of Power Budgeting Data are selected by increasing index values. The default value of this register is undefined.

7.8.1.3 Power Budgeting Control Register (Offset 06h) §

The Power Budgeting Control Register permits system software to enable extended power budgeting and to grant additional power to a Device above that defined by default for the associated form-factor.

§ Figure 7-138 details allocation of register fields in the Aux Power Control register; § Table 7-122 provides the respective bit definitions.

Base 6.4 vs Base 6.3

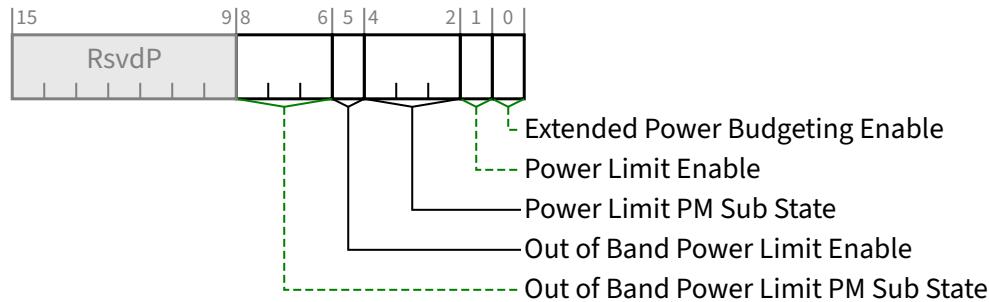


Figure 7-138 Power Budgeting Control Register §

Table 7-122 Power Budgeting Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 0 | <p>Extended Power Budgeting Enable – If Set, Power Budgeting is permitted to return non-zero values in the Power Budgeting Data Register bits 31:21. If Clear, those bits must return all zeros for all values of the Power Budgeting Data Select Register.</p> <p>If Set, the Power Budgeting Sense Detect Register is permitted to return non-zero values. If Clear, that register must return all zeros.</p> <p>This bit is hardwired to 0b when <u>Extended Power Budgeting Supported</u> is Clear.</p> <p>Default is zero.</p> | RW |
| 1 | <p>Power Limit Enable – If Set, the Power Limit PM Sub State field is meaningful.</p> <p>The value of this field in the lowest numbered Function with <u>Power Limit Supported</u> Set applies to all Functions of the Device. When present, the value of this bit in all other Functions is ignored by hardware.</p> <p>When <u>Power Limit Supported</u> is Clear, this bit is permitted to be hardwired to zero.</p> <p>It is recommended that system software / firmware configure this field identically in all Functions. Doing so provides a standard mechanism for a device driver to understand its Function's power configuration.</p> <p>Default is zero.</p> | RWS / RsvdP |
| 4:2 | <p>Power Limit PM Sub State – If Power Limit Enable is Set, this field, in conjunction with the <u>Out of Band Power Limit Enable</u> and <u>Out of Band Power Limit PM Sub State</u> fields, indicates the PM Sub State used by the Device.</p> <p>The value of this field in the lowest numbered Function with <u>Power Limit Supported</u> Set applies to all Functions of the Device. When present, the value of this field in all other Functions is ignored by hardware.</p> <p>When <u>Power Limit Supported</u> is Clear, this field is permitted to be hardwired to zero.</p> <p>It is recommended that system software / firmware configure this field identically in all Functions. Doing so provides a standard mechanism for a device driver to understand its Function's power configuration.</p> <p>Default is zero.</p> | RWS / RsvdP |
| 5 | <p>Out of Band Power Limit Enable – If Set, the Out of Band Power Limit PM Sub State field is meaningful.</p> <p>When this field is present, all Functions of the Device must contain the same value.</p> <p>When <u>Power Limit Supported</u> is Clear, this bit is permitted to be hardwired to zero.</p> | HwInit / RsvdP |

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| | <p>It is permitted that this field change after the Function is Configuration Ready. This could happen, for example, if this field is configured via MCTP over PCIe. Mechanisms used to delay access to this field until it is meaningful are outside the scope of this specification (e.g., using the SFI mechanism or using _DSM calls to grant system software access to the fields).</p> <p>Default is zero.</p> | |
| 8:6 | <p>Out of Band Power Limit PM Sub State – If Out of Band Power Limit Enable is Set, this field, in conjunction with the Power Limit Enable and Power Limit PM Sub State fields, indicates the PM Sub State used by the Device.</p> <p>When this field is present, all Functions of the Device must contain the same value.</p> <p>When Power Limit Supported is Clear, this bit is permitted to be hardwired to zero.</p> <p>It is permitted that this field change after the Function is Configuration Ready. This could happen, for example, if this field is configured via MCTP over PCIe. Mechanisms used to delay access to this field until it is meaningful are outside the scope of this specification (e.g., using the SFI mechanism or using _DSM calls to grant system software access to the fields).</p> <p>Default is zero.</p> | HwInit / RsvdP |

7.8.1.4 Power Budgeting Data Register (Offset 08h) §

This read-only register returns the DWORD of Power Budgeting Data selected by the Power Budgeting 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 Power Budgeting Data Select Register causes information for a different operating condition to be returned. If the Power Budgeting Data Select Register contains a value greater than or equal to the number of operating conditions for which the device provides power information, this register must return all zeros. The default value of this register is undefined. § Figure 7-139 details allocation of register fields in the Power Budgeting Data Register ; § Table 7-123 provides the respective bit definitions.

In earlier versions of this specification, bits 31:21 of this register were RsvdP. In order to ensure that the new uses of these bits do not confuse existing software:

- Extended Power Budgeting entries are hidden when Extended Power Budgeting Enable is Clear (the default). When Extended Power Budgeting Enable is Clear and Power Budgeting Data Select selects an Extended Power Budgeting entry, the Data register must return 0000 0000h.
- Extended Power Budgeting Data entries must be located after non-extended Power Budgeting Data entries (i.e., all entries where bits 31:21 are zero must use a smaller Power Budgeting Data Select value than any entry where bits 31:21 are non-zero).

The Base Power and Data Scale fields describe the power usage of the device; the Power Rail , Type , PM State , and PM Sub State fields describe the conditions under which the device has this power usage.

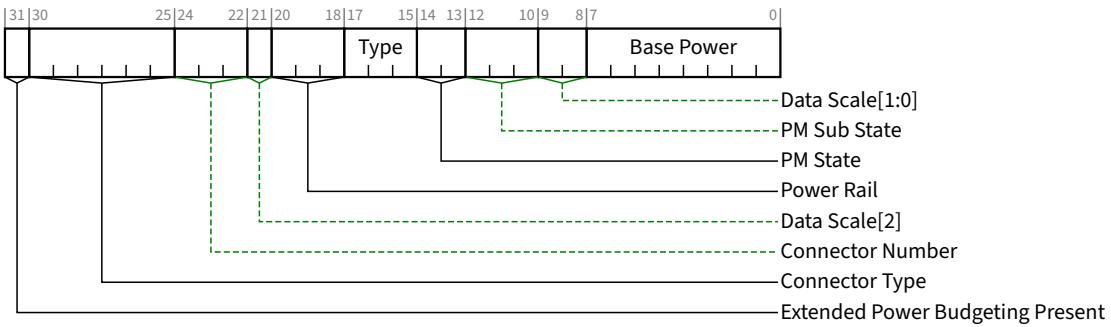


Figure 7-139 Power Budgeting Data Register §

Table 7-123 Power Budgeting Data Register §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|--|-------------|--------------------------------------|-------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--------------------------------------|------------|--|----|
| 7:0 | <p>Base Power - Specifies in watts the base power value in the given operating condition. This value must be multiplied by the data scale to produce the actual power consumption value except if Extended Power Budgeting Enable is Clear, the Data Scale [1:0] field equals 00b (1.0x) and Base Power exceeds EFh, the following alternative encodings are used:</p> <table> <tbody> <tr><td>F0h</td><td>> 239 W and ≤ 250 W Slot Power Limit</td></tr> <tr><td>F1h</td><td>> 250 W and ≤ 275 W Slot Power Limit</td></tr> <tr><td>F2h</td><td>> 275 W and ≤ 300 W Slot Power Limit</td></tr> <tr><td>F3h</td><td>> 300 W and ≤ 325 W Slot Power Limit</td></tr> <tr><td>F4h</td><td>> 325 W and ≤ 350 W Slot Power Limit</td></tr> <tr><td>F5h</td><td>> 350 W and ≤ 375 W Slot Power Limit</td></tr> <tr><td>F6h</td><td>> 375 W and ≤ 400 W Slot Power Limit</td></tr> <tr><td>F7h</td><td>> 400 W and ≤ 425 W Slot Power Limit</td></tr> <tr><td>F8h</td><td>> 425 W and ≤ 450 W Slot Power Limit</td></tr> <tr><td>F9h</td><td>> 450 W and ≤ 475 W Slot Power Limit</td></tr> <tr><td>FAh</td><td>> 475 W and ≤ 500 W Slot Power Limit</td></tr> <tr><td>FBh</td><td>> 500 W and ≤ 525 W Slot Power Limit</td></tr> <tr><td>FCh</td><td>> 525 W and ≤ 550 W Slot Power Limit</td></tr> <tr><td>FDh</td><td>> 550 W and ≤ 575 W Slot Power Limit</td></tr> <tr><td>FEh</td><td>> 575 W and ≤ 600 W Slot Power Limit</td></tr> <tr><td>FFh</td><td>Reserved for values greater than 600 W</td></tr> </tbody> </table> | F0h | > 239 W and ≤ 250 W Slot Power Limit | F1h | > 250 W and ≤ 275 W Slot Power Limit | F2h | > 275 W and ≤ 300 W Slot Power Limit | F3h | > 300 W and ≤ 325 W Slot Power Limit | F4h | > 325 W and ≤ 350 W Slot Power Limit | F5h | > 350 W and ≤ 375 W Slot Power Limit | F6h | > 375 W and ≤ 400 W Slot Power Limit | F7h | > 400 W and ≤ 425 W Slot Power Limit | F8h | > 425 W and ≤ 450 W Slot Power Limit | F9h | > 450 W and ≤ 475 W Slot Power Limit | FAh | > 475 W and ≤ 500 W Slot Power Limit | FBh | > 500 W and ≤ 525 W Slot Power Limit | FCh | > 525 W and ≤ 550 W Slot Power Limit | FDh | > 550 W and ≤ 575 W Slot Power Limit | FEh | > 575 W and ≤ 600 W Slot Power Limit | FFh | Reserved for values greater than 600 W | RO |
| F0h | > 239 W and ≤ 250 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F1h | > 250 W and ≤ 275 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F2h | > 275 W and ≤ 300 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F3h | > 300 W and ≤ 325 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F4h | > 325 W and ≤ 350 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F5h | > 350 W and ≤ 375 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F6h | > 375 W and ≤ 400 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F7h | > 400 W and ≤ 425 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F8h | > 425 W and ≤ 450 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F9h | > 450 W and ≤ 475 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FAh | > 475 W and ≤ 500 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FBh | > 500 W and ≤ 525 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FCh | > 525 W and ≤ 550 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDh | > 550 W and ≤ 575 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FEh | > 575 W and ≤ 600 W Slot Power Limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FFh | Reserved for values greater than 600 W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9:8 | <p>Data Scale[1:0] - Specifies the scale to apply to the Base Power value. The power consumption of the device is determined by multiplying the contents of the Base Power field with the value corresponding to the encoding returned by this field, except as noted above.</p> <p>Note that <u>Data Scale [2]</u> and <u>Data Scale [1:0]</u> are not contiguous within this register.</p> <p>Defined encodings are:</p> <table> <tbody> <tr><td>000b</td><td>1.0x</td></tr> <tr><td>001b</td><td>0.1x</td></tr> </tbody> </table> | 000b | 1.0x | 001b | 0.1x | RO | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000b | 1.0x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001b | 0.1x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------------|---|-------------|--|--------------------|--|-------------|--|-------------|-----------------|-------------|--|-------------|--|-------------|---------------|---------------|-----------------------------------|----|
| | <p>010b 0.01x</p> <p>011b 0.001x</p> <p>100b 10x</p> <p>101b 100x</p> <p>Others Reserved</p> | | | | | | | | | | | | | | | | | |
| 12:10 | <p>PM Sub State - Specifies the power management sub state of the operating condition being described.</p> <p>Defined encodings are:</p> <table> <tr> <td>000b</td> <td>Default Sub State</td> </tr> <tr> <td>001b - 111b</td> <td>Device Specific Sub State</td> </tr> </table> | 000b | Default Sub State | 001b - 111b | Device Specific Sub State | RO | | | | | | | | | | | | |
| 000b | Default Sub State | | | | | | | | | | | | | | | | | |
| 001b - 111b | Device Specific Sub State | | | | | | | | | | | | | | | | | |
| 14:13 | <p>PM State - Specifies the power management state of the operating condition being described.</p> <p>Defined encodings are:</p> <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> <p>A device returns 11b in this field and Auxiliary or PME Aux in the Type field to specify the <u>D3Cold PM State</u>. An encoding of 11b along with any other Type field value specifies the <u>D3Hot state</u>.</p> | 00b | D0 | 01b | D1 | 10b | D2 | 11b | D3 | RO | | | | | | | | |
| 00b | D0 | | | | | | | | | | | | | | | | | |
| 01b | D1 | | | | | | | | | | | | | | | | | |
| 10b | D2 | | | | | | | | | | | | | | | | | |
| 11b | D3 | | | | | | | | | | | | | | | | | |
| 17:15 | <p>Type - Specifies the type of the operating condition being described. Defined encodings are:</p> <table> <tr> <td>000b</td> <td>PME Aux -- <u>Sustained Power consumed in D3Cold when PME_En is Set and Aux Power PM Enable is Clear</u></td> </tr> <tr> <td>001b</td> <td>Auxiliary -- <u>Sustained Power consumed in D3Cold when Aux Power PM Enable is Set</u></td> </tr> <tr> <td>010b</td> <td>Idle -- <u>Sustained Power consumed when the Function or Device has been idle for 20 seconds or more</u></td> </tr> <tr> <td>011b</td> <td>Sustained Power</td> </tr> <tr> <td>100b</td> <td>Sustained Power in Emergency Power Reduction State (see § Section 6.24)</td> </tr> <tr> <td>101b</td> <td>Maximum Power in Emergency Power Reduction State (see § Section 6.24)</td> </tr> <tr> <td>111b</td> <td>Maximum Power</td> </tr> <tr> <td>Others</td> <td>All other encodings are Reserved.</td> </tr> </table> <p>The following measurement definitions apply to this field unless the form-factor specification explicitly states otherwise:</p> <ul style="list-style-type: none"> • Sustained Power means the power consumed when the Device is performing at its maximum throughput, measured as an average over 1 second. • Maximum Power means the power consumed when the Device is performing at its maximum throughput, measured over a 100 µs moving window. Note that Maximum Power can easily exceed sustained power by as much as 250%. Maximum Power consumption is frequently associated with changes in Function D-state. | 000b | PME Aux -- <u>Sustained Power consumed in D3Cold when PME_En is Set and Aux Power PM Enable is Clear</u> | 001b | Auxiliary -- <u>Sustained Power consumed in D3Cold when Aux Power PM Enable is Set</u> | 010b | Idle -- <u>Sustained Power consumed when the Function or Device has been idle for 20 seconds or more</u> | 011b | Sustained Power | 100b | Sustained Power in Emergency Power Reduction State (see § Section 6.24) | 101b | Maximum Power in Emergency Power Reduction State (see § Section 6.24) | 111b | Maximum Power | Others | All other encodings are Reserved. | RO |
| 000b | PME Aux -- <u>Sustained Power consumed in D3Cold when PME_En is Set and Aux Power PM Enable is Clear</u> | | | | | | | | | | | | | | | | | |
| 001b | Auxiliary -- <u>Sustained Power consumed in D3Cold when Aux Power PM Enable is Set</u> | | | | | | | | | | | | | | | | | |
| 010b | Idle -- <u>Sustained Power consumed when the Function or Device has been idle for 20 seconds or more</u> | | | | | | | | | | | | | | | | | |
| 011b | Sustained Power | | | | | | | | | | | | | | | | | |
| 100b | Sustained Power in Emergency Power Reduction State (see § Section 6.24) | | | | | | | | | | | | | | | | | |
| 101b | Maximum Power in Emergency Power Reduction State (see § Section 6.24) | | | | | | | | | | | | | | | | | |
| 111b | Maximum Power | | | | | | | | | | | | | | | | | |
| Others | All other encodings are Reserved. | | | | | | | | | | | | | | | | | |
| 20:18 | <p>Power Rail - Specifies the thermal load or power rail of the operating condition being described.</p> <p>Defined encodings are:</p> <table> <tr> <td>000b</td> <td>Power (12V)</td> </tr> </table> | 000b | Power (12V) | RO | | | | | | | | | | | | | | |
| 000b | Power (12V) | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------|---|-----------------|------------------------------------|-----------------|--|-----------------|---|-----------------|---|-----------------|-------------------|-----------------|--|-----------------|--|-----------------|--|-----------------|--|-----------------|--|-----------------|--|-----------------|--|-----------------|--|-----------------|--|-----------------|--|----|
| | <p>001b Power (3.3V)</p> <p>010b Power (1.5V or 1.8V)</p> <p>100b Power (48V)</p> <p>101b Power (5V)</p> <p>111b Thermal</p> <p>Others All other encodings are Reserved.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | <p>Data Scale[2] – Upper bit of Data Scale field. See <u>Data Scale [1:0]</u> for details.</p> <p>This bit must be zero if <u>Extended Power Budgeting Enable</u> is Clear.</p> | RO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24:22 | <p>Connector Number – Up to 8 connectors that supply power are supported on an add-in card (including the edge connector(s)). This field indicates which power connector is associated with this entry.</p> <p>If <u>Power Budgeting Sense Detect Supported</u> is Set, an instance of this field must be implemented for every power connector that the add-in card supports.</p> <p>Connector Numbers represent a single physical connector and are global across the add-in card. Connector Numbers must be consistent across all Functions in a Multi-Function Device. Connector Numbers must match when a single connector contains more than one power rail or when a single connector is associated with more than one <u>Connector Type</u> (e.g., Types 00 0110b and 00 0111b).</p> <p>Connector Number values and the mapping of Connector Number to physical location are outside the scope of this specification. For form-factor specifications that specify connector placement, it is recommended that those specifications define connector numbering based on placement rules.</p> <p>This field must be zero if <u>Extended Power Budgeting Enable</u> is Clear.</p> <p>Software must ignore the value in this field if <u>Extended Power Budgeting Present</u> is Clear.</p> | RO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30:25 | <p>Connector Type – Indicates the connector type. If <u>Power Budgeting Sense Detect Supported</u> is Set, an instance of this field must be implemented for every power connector that the adaptor supports. Values are:</p> <table> <tbody> <tr> <td>00 0000b</td><td>Form-factor defined edge connector</td></tr> <tr> <td>00 0001b</td><td>Non-Connector power provided by the system (e.g., soldered down)</td></tr> <tr> <td>00 0010b</td><td>Non-Connector power provided by the option card (e.g., battery)</td></tr> <tr> <td>00 0011b</td><td>Non-Connector power not provided by the system or the option card (e.g., power supplied by an external chassis)</td></tr> <tr> <td>00 0100b</td><td>CEM 2x3 connector</td></tr> <tr> <td>00 0101b</td><td>CEM 2x4 connector with either 2x3 cable or 2x4 cable (see below)</td></tr> <tr> <td>00 0110b</td><td>CEM 2x4 connector with 2x3 cable (see below)</td></tr> <tr> <td>00 0111b</td><td>CEM 2x4 connector with 2x4 cable (see below)</td></tr> <tr> <td>00 1000b</td><td>CEM 12VHPWR connector, cable has both Sense0 and Sense1 Open</td></tr> <tr> <td>00 1001b</td><td>CEM 12VHPWR connector, cable has Sense0 Grounded and Sense1 Open</td></tr> <tr> <td>00 1010b</td><td>CEM 12VHPWR connector, has Sense0 Open and Sense1 Grounded</td></tr> <tr> <td>00 1011b</td><td>CEM 12VHPWR connector, has both Sense0 and Sense1 Grounded</td></tr> <tr> <td>00 1100b</td><td>CEM 48VHPWR connector, cable has both Sense0 and Sense1 Open</td></tr> <tr> <td>00 1101b</td><td>CEM 48VHPWR connector, cable has Sense0 Grounded and Sense1 Open</td></tr> <tr> <td>00 1110b</td><td>CEM 48VHPWR connector, cable has Sense0 Open and Sense1 Grounded</td></tr> </tbody> </table> | 00 0000b | Form-factor defined edge connector | 00 0001b | Non-Connector power provided by the system (e.g., soldered down) | 00 0010b | Non-Connector power provided by the option card (e.g., battery) | 00 0011b | Non-Connector power not provided by the system or the option card (e.g., power supplied by an external chassis) | 00 0100b | CEM 2x3 connector | 00 0101b | CEM 2x4 connector with either 2x3 cable or 2x4 cable (see below) | 00 0110b | CEM 2x4 connector with 2x3 cable (see below) | 00 0111b | CEM 2x4 connector with 2x4 cable (see below) | 00 1000b | CEM 12VHPWR connector, cable has both Sense0 and Sense1 Open | 00 1001b | CEM 12VHPWR connector, cable has Sense0 Grounded and Sense1 Open | 00 1010b | CEM 12VHPWR connector, has Sense0 Open and Sense1 Grounded | 00 1011b | CEM 12VHPWR connector, has both Sense0 and Sense1 Grounded | 00 1100b | CEM 48VHPWR connector, cable has both Sense0 and Sense1 Open | 00 1101b | CEM 48VHPWR connector, cable has Sense0 Grounded and Sense1 Open | 00 1110b | CEM 48VHPWR connector, cable has Sense0 Open and Sense1 Grounded | RO |
| 00 0000b | Form-factor defined edge connector | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0001b | Non-Connector power provided by the system (e.g., soldered down) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0010b | Non-Connector power provided by the option card (e.g., battery) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0011b | Non-Connector power not provided by the system or the option card (e.g., power supplied by an external chassis) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0100b | CEM 2x3 connector | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0101b | CEM 2x4 connector with either 2x3 cable or 2x4 cable (see below) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0110b | CEM 2x4 connector with 2x3 cable (see below) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 0111b | CEM 2x4 connector with 2x4 cable (see below) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1000b | CEM 12VHPWR connector, cable has both Sense0 and Sense1 Open | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1001b | CEM 12VHPWR connector, cable has Sense0 Grounded and Sense1 Open | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1010b | CEM 12VHPWR connector, has Sense0 Open and Sense1 Grounded | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1011b | CEM 12VHPWR connector, has both Sense0 and Sense1 Grounded | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1100b | CEM 48VHPWR connector, cable has both Sense0 and Sense1 Open | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1101b | CEM 48VHPWR connector, cable has Sense0 Grounded and Sense1 Open | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 1110b | CEM 48VHPWR connector, cable has Sense0 Open and Sense1 Grounded | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>00 1111b CEM 48VHPWR connector, cable has both Sense0 and Sense1 Grounded</p> <p>01 0000b CEM 12V-2x6: 150 W or higher cable present</p> <p>01 0001b CEM 12V-2x6: 300 W or higher cable present</p> <p>01 0010b CEM 12V-2x6: 450 W or higher cable present</p> <p>01 0011b CEM 12V-2x6: 600 W cable present</p> <p>01 0100b to 10 1111b Reserved for PCI-SIG use</p> <p>11 0000b to 11 1111b Vendor Specific Power Connectors</p> <p>Each CEM 2x4 connector must have either one entry with Connector Type 00 0101b or two entries with Connector Types 00 0110b and 00 0111b.</p> <p>Each 12V-2x6 connector must have an entry for the lowest power cable it supports. Each 12V-2x6 connector must have an entry for every cable that has a different power consumption value.</p> <p>This field must be zero if <u>Extended Power Budgeting Enable</u> is Clear.</p> <p>Software must ignore the value in this field if <u>Extended Power Budgeting Present</u> is Clear.</p> | |
| 31 | <p>Extended Power Budgeting Present – Indicates that bits 30:22 contain Extended Power Budgeting Data. This bit must be 0b if <u>Extended Power Budgeting Enable</u> is Clear.</p> | RO |

Except for Type = 000b and 001b, Power Budgeting data for the same operating condition and PM Sub State values represent simultaneous consumption. Functions must report a complete set of Power Budgeting data for each supported operating condition and PM Sub State combination.

Power Budgeting data with different PM Sub State values represent mutually exclusive consumption. For a given operating condition, a Function is in exactly one PM Sub State. When Power Limit Supported is Clear, implementation specific mechanisms are used to determine the current PM Sub State.

A device that implements the Power Budgeting Extended Capability is required to provide data values for D0 Maximum and D0 Sustained PM State and Type combinations for every power rail from which it consumes power; data for the D0 Maximum and D0 Sustained for Thermal must also be provided if these values are different from the sum of the values for an operating condition reported for D0 Maximum and D0 Sustained on the power rails.

Devices that support auxiliary power or PME from auxiliary power must provide data for the appropriate power Type (Auxiliary or PME Aux) on the appropriate Power Rail(s).

- If the reported PME Aux or Auxiliary value is greater than the default for the associated form-factor, the Function is limited to the form-factor values unless either PME_En or Aux Power PM Enable are Set.
- The PME Aux and Auxiliary entries are mutually exclusive. The values of PME_En and Aux Power PM Enable determine which entries are meaningful.

If the reported PME Aux or Auxiliary value is greater than the Aux_Current, the Function is limited by Aux_Current unless Aux Power PM Enable is Set and one of the following is true:

- Power Limit Enable is Set,
- Out of Band Power Limit Enable is Set, or
- the Request D3_Cold Aux Power Limit _DSM call was used to request additional power (for details, see [Firmware]).

If a device implements Emergency Power Reduction State, it must report Power Budgeting values for the following:

- Maximum Emergency Power Reduction State, PM State D0, all power rails used by the device

- Maximum Emergency Power Reduction State, PM State D0, Thermal (if different from the sum of the preceding values)
- Sustained Emergency Power Reduction State, PM State D0, all power rails used by the device
- Sustained Emergency Power Reduction State, PM State : D0, Thermal (if different from the sum of the preceding values)

7.8.1.5 Power Budgeting Capability Register (Offset 0Ch) §

This register indicates the power budgeting capabilities of a device. § Figure 7-140 details allocation of register fields in the Power Budgeting Capability Register ; § Table 7-124 provides the respective bit definitions.

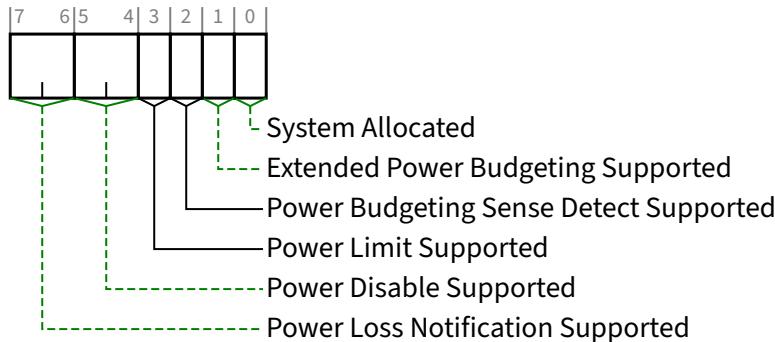


Figure 7-140 Power Budgeting Capability Register §

Table 7-124 Power Budgeting Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|--|---------------|
| 0 | System Allocated - When Set, this bit indicates that the power budget for the device is included within the system power budget. Reported Power Budgeting Data for this device must be ignored by software for power budgeting decisions if this bit is Set. | <u>HwInit</u> |
| 1 | Extended Power Budgeting Supported – If Set, the Extended Power Budgeting Enable bit is meaningful. | <u>HwInit</u> |
| 2 | Power Budgeting Sense Detect Supported – If Set, the Power Budgeting Sense Detect Register is meaningful. This bit must be Clear if Extended Power Budgeting Supported is Clear. | <u>HwInit</u> |
| 3 | Power Limit Supported – If Set, the Power Limit Enable , Power Limit PM Sub State , Out of Band Power Limit Enable , and Out of Band Power Limit PM Sub State fields are meaningful. This bit must be Clear if Extended Power Budgeting Supported is Clear. | <u>HwInit</u> |
| 5:4 | Power Disable Supported – Indicates the supported use model for optional form-factor defined Power Disable functionality. Encodings are: 00b Power Disable support not reported 01b Power Disable is supported for removal of main power. The timings associated with this mode are optimized to support the use of Power Disable to recover a non-responsive device. | <u>HwInit</u> |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>10b Power Disable with abbreviated assertion time is supported for removal of main power. The timings associated with this mode are optimized to using Power Disable to request the Device enter or exit D3 Cold .</p> <p>11b Reserved</p> <p>For Multi-Function Devices associated with an Upstream Port, all Functions that contain this field must return the same value.</p> | |
| 7:6 | <p>Power Loss Notification Supported – This field indicates Device support for the optional Power Loss Notification feature. Power Loss Notification is an optional form-factor feature that permits the platform to inform an add-in card of an upcoming loss of power and optionally for the add-in card to indicate that it is ready for that power loss. In the M.2 form-factor, Power Loss Notification uses the optional PLN# signal and Power Loss Acknowledgment uses the optional PLA_S2# and PLA_S3# signals. Other form-factors may define different mechanisms.</p> <p>Encodings of this field are:</p> <ul style="list-style-type: none"> 00b Power Loss Notification support not reported 01b Power Loss Notification supported Power Loss Acknowledgement not supported 10b Power Loss Notification supported Power Loss Acknowledgement supported 11b Reserved <p>For Multi-Function Devices associated with an Upstream Port, all Functions that contain this field must return the same value.</p> | HwInit |

7.8.1.6 Power Budgeting Sense Detect Register (Offset 0Dh) §

Whenever the adapter is receiving any power, this register reports, for each implemented power connector, which sense wires are currently detected.

Any adapter that implements a Power Budgeting Extended Capability with Power Budgeting Sense Detect Supported Set, must provide Connector Sense Detect fields for each connector that it supports, and must hardwire the fields for unsupported connectors to all zeros.

This register is RsvdP if Power Budgeting Sense Detect Supported is Clear. This register must return all zeros if Extended Power Budgeting Enable is Clear.

This register is only meaningful in the lowest numbered Function that contains the Power Budgeting Extended Capability . This register is undefined in all other Functions even if the Power Budgeting Sense Detect Supported is Set.

§ Figure 7-141 details allocation of register fields in the Power Budgeting Sense Detect Register ; § Table 7-125 provides the respective bit definitions. § Table 7-126 defines the encodings based on Connector Type values.

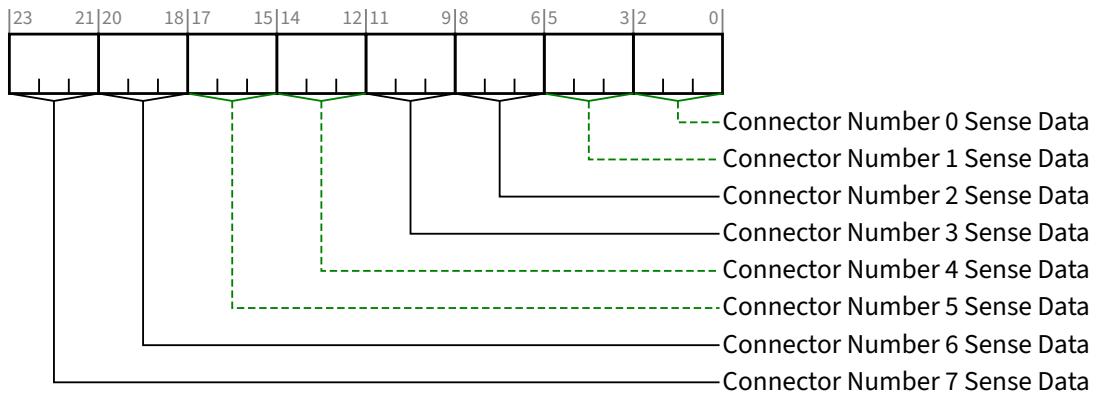


Figure 7-141 Power Budgeting Sense Detect Register §

Table 7-125 Power Budgeting Sense Detect Register §

| Bit Location | Register Description | Attributes |
|--------------|--------------------------------------|------------|
| 2:0 | Connector Number 0 Sense Data | RO |
| 5:3 | Connector Number 1 Sense Data | RO |
| 8:6 | Connector Number 2 Sense Data | RO |
| 11:9 | Connector Number 3 Sense Data | RO |
| 14:12 | Connector Number 4 Sense Data | RO |
| 17:15 | Connector Number 5 Sense Data | RO |
| 20:18 | Connector Number 6 Sense Data | RO |
| 23:21 | Connector Number 7 Sense Data | RO |

Table 7-126 Power Budgeting Sense Detect Encodings §

| ↓↓Connector Type↓↓ | | Encoding |
|----------------------------|-------------------|--|
| 00 0000b to 00 0011b | Bit 0 | Main Power Detected. If Auxiliary power is not used, this bit is permitted to be hardwired to 1b. |
| | Bit 1 | Aux Power Detected. If auxiliary power is not used, this bit is permitted to be hardwired to 0b. When no dedicated Aux Power pin(s) are defined in the implemented form-factor, this bit has a form-factor specific meaning. |
| | Bit 2 | Form-factor specific meaning |
| 00 0100b | CEM 2x3 connector | |
| | 000b | Cable is not present, Sense not detected |
| | 001b | Cable is present, Sense detected |
| | Others | Reserved |

Base 6.4 vs Base 6.3

| ↓↓Connector Type↓ | ↑↑Connector Type↑ | Encoding |
|----------------------------|-------------------|--|
| | | CEM 2x4 connector |
| 00 0101b to 00 0111b | | <p>000b Cable is not present, both Sense0 and Sense1 not detected</p> <p>001b 2x3 cable is present, Sense0 detected, Sense1 not detected</p> <p>010b Reserved condition, Sense0 not detected, Sense1 detected</p> <p>011b 2x4 cable is present, both Sense0 and Sense1 detected</p> <p>Others Reserved</p> |
| | | CEM 12VHPWR connector |
| 00 1000b to 00 1011b | | <p>0xxb Cable is not present</p> <p>100b 12VHPWR cable is present, both Sense0 and Sense1 are Open</p> <p>101b 12VHPWR cable is present, Sense0 is Grounded, Sense1 Open</p> <p>110b 12VHPWR cable is present, Sense0 Open, Sense1 Grounded</p> <p>111b 12VHPWR cable is present, both Sense0 and Sense1 Grounded</p> <p>Others Reserved</p> |
| | | CEM 48VHPWR connector |
| 00 1100b to 00 1111b | | <p>0xxb Cable is not present</p> <p>100b 48VHPWR cable is present, both Sense0 and Sense1 Open</p> <p>101b 48VHPWR cable is present, Sense0 Grounded, Sense1 Open</p> <p>110b 48VHPWR cable is present, Sense0 Open, Sense1 Grounded</p> <p>111b 48VHPWR cable is present, both Sense0 and Sense1 Grounded</p> <p>Others Reserved</p> |
| | | CEM 12V-2x6 connector, 150 W or higher cable |
| 01 0000b | | <p>000b Cable is not present</p> <p>100b 150 W 12V-2x6 cable is present</p> <p>101b 300 W 12V-2x6 cable is present</p> <p>110b 450 W 12V-2x6 cable is present</p> <p>111b 600 W 12V-2x6 cable is present</p> <p>Others Reserved</p> |
| | | CEM 12V-2x6 connector, 300 W or higher cable |
| 01 0001b | | <p>000b Cable is not present</p> <p>100b Cable is not present or 150 W 12V-2x6 cable is present¹⁸⁸</p> <p>101b 300 W 12V-2x6 cable is present</p> <p>110b 450 W 12V-2x6 cable is present</p> <p>111b 600 W 12V-2x6 cable is present</p> <p>Others Reserved</p> |

188. Detecting the 12V-2x6 150 W cable requires additional circuitry. This circuitry is optional when the add-in card always requires more than 150 W from the 12V-2x6 connector.

| ↓↓Connector Type↓ ↑↑Connector Type↑ | Encoding |
|--|---|
| 01 0010b | <p>CEM 12V-2x6 connector, 450 W or higher cable</p> <p>000b Cable is not present</p> <p>100b Cable is not present or 150 W 12V-2x6 cable is present¹⁸⁹</p> <p>101b 300 W 12V-2x6 cable is present¹⁹⁰</p> <p>110b 450 W 12V-2x6 cable is present</p> <p>111b 600 W 12V-2x6 cable is present</p> <p>Others Reserved</p> |
| 01 0011b | <p>CEM 12V-2x6 connector, 600 W cable</p> <p>000b Cable is not present</p> <p>100b Cable is not present or 150 W 12V-2x6 cable is present¹⁹¹</p> <p>101b 300 W 12V-2x6 cable is present¹⁹²</p> <p>110b 450 W 12V-2x6 cable is present¹⁹³</p> <p>111b 600 W 12V-2x6 cable is present</p> <p>Others Reserved</p> |
| 01 0100b to 10 1111h | Reserved for PCI-SIG use |
| 11 0000b to 11 1111b | Vendor Specific |

7.8.2 Latency Tolerance Reporting (LTR) Extended Capability §

The PCI Express Latency Tolerance Reporting (LTR) Extended Capability is an optional Extended Capability that allows software to provide platform latency information to components with Upstream Ports (Endpoints and Switches), and is required for Switch Upstream Ports and Endpoints if the Function supports the LTR mechanism. It is not applicable to Root Ports, Bridges, or Switch Downstream Ports.

For a Multi-Function Device associated with the Upstream Port of a component that implements the LTR mechanism, this Capability structure must be implemented only in Function 0, and must control the component's Link behavior on behalf of all the Functions of the Device.

RCiEPs implemented as Multi-Function Devices are permitted to implement this Capability structure in more than one Function of the Multi-Function Device .

189. Detecting the 12V-2x6 150 W cable requires additional circuitry. This circuitry is optional when the add-in card always requires more than 150 W from the 12V-2x6 connector.

190. This combination indicates the provided power is less than the connector requirements. The add-in card does not use this cable. This sense data encoding is used to tell software.

191. Detecting the 12V-2x6 150 W cable requires additional circuitry. This circuitry is optional when the add-in card always requires more than 150 W from the 12V-2x6 connector.

192. This combination indicates the provided power is less than the connector requirements. The add-in card does not use this cable. This sense data encoding is used to tell software.

193. This combination indicates the provided power is less than the connector requirements. The add-in card does not use this cable. This sense data encoding is used to tell software.

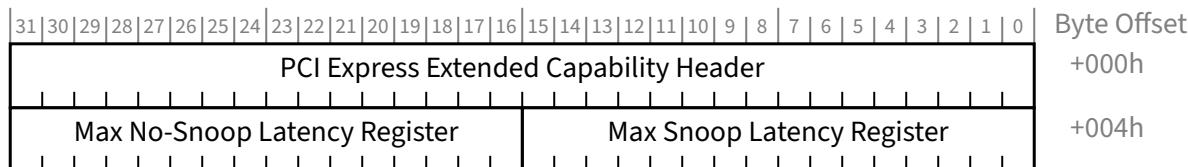


Figure 7-142 LTR Extended Capability Structure §

7.8.2.1 LTR Extended Capability Header (Offset 00h) §

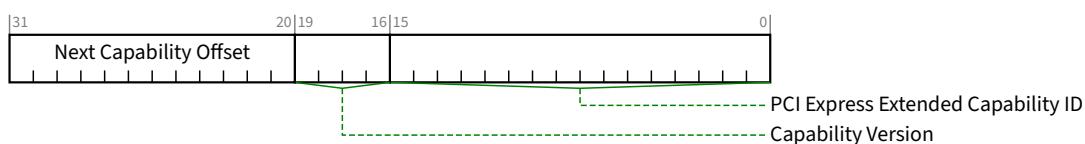


Figure 7-143 LTR Extended Capability Header §

Table 7-127 LTR Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability for the LTR Extended Capability is 0018h. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. LTR Capabilities Register is implemented and must be 1h otherwise. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

ECN: Base 6.3
LTR-MFD△

7.8.2.2 Max Snoop Latency Register (Offset 04h) §

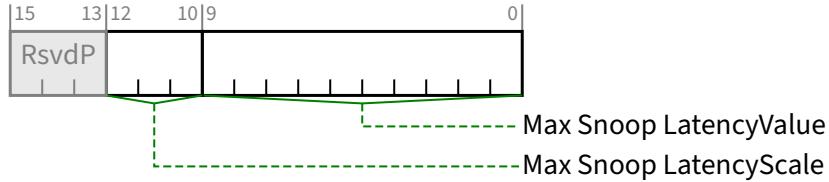


Figure 7-144 Max Snoop Latency Register §

Table 7-128 Max Snoop Latency Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 9:0 | <p>Max Snoop LatencyValue - Along with the Max Snoop LatencyScale field, this register specifies the maximum snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms.</p> <p>The default value for this field is 00 0000 0000b.</p> | RW |
| 12:10 | <p>Max Snoop LatencyScale - This register provides a scale for the value contained within the Max Snoop LatencyValue field. Encoding is the same as the LatencyScale fields in the LTR Message. See § Section 6.18 . It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms.</p> <p>The default value for this field is 000b.</p> <p>Hardware operation is undefined if software writes a Not Permitted value to this field.</p> | RW |

7.8.2.3 Max No-Snoop Latency Register (Offset 06h) §

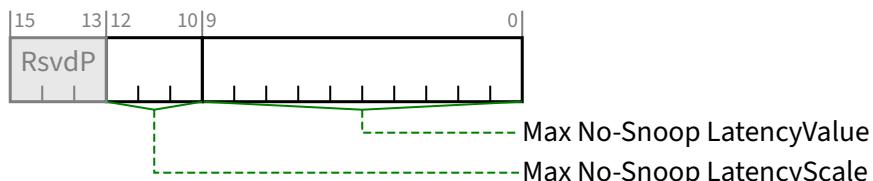


Figure 7-145 Max No-Snoop Latency Register §

Table 7-129 Max No-Snoop Latency Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 9:0 | <p>Max No-Snoop LatencyValue - Along with the Max No-Snoop LatencyScale field, this register specifies the maximum no-snoop latency that a device is permitted to request. Software should set this to the</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>platform's maximum supported latency or less. It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms.</p> <p>The default value for this field is 00 0000 0000b.</p> | |
| 12:10 | <p>Max No-Snoop LatencyScale - This register provides a scale for the value contained within the Max No-Snoop LatencyValue field. Encoding is the same as the LatencyScale fields in the LTR Message. See § Section 6.18 . It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms.</p> <p>The default value for this field is 000b.</p> <p>Hardware operation is undefined if software writes a Not Permitted value to this field.</p> | RW |

7.8.2.4 [LTR Capabilities Register \(Offset 08h\)](#) §

ECN: Base 6.3
LTR-MFD△◀

↑↑Hardware must implement this register if the Capability Version in the LTR Extended Capability Header register is 2h or greater. This register is not present if the Capability Version is 1h.↑



Figure 7-146 ECN: Base 6.3 LTR-MFD LTR Capabilities Register §

Table 7-130 [LTR Capabilities Register](#) §

| Bit Location | Register Description | Attributes |
|--------------|---|----------------------|
| ↑↑1↑ | <p>LTR Enable Preserved Across FLR – When Set, the LTR Mechanism Enable bit in Function 0 is not affected by a Function Level Reset (FLR) to Function 0. When Clear or when the LTR Capabilities Register is not present, the LTR Mechanism Enable bit in Function 0 is affected by FLR ↑</p> <p>↑↑This field is RsvdP if the Function Level Reset Capability bit is Clear.↑</p> | ↑↑HwInit / RsvdP↑ |

7.8.3 L1 PM Substates Extended Capability §

The L1 PM Substates Extended Capability is an optional Extended Capability, that is required if L1 PM Substates is implemented at a Port. The L1 PM Substates Extended Capability structure is defined as shown in § Figure 7-147 .

For a Multi-Function Device associated with an Upstream Port implementing L1 PM Substates, this Extended Capability Structure must be implemented only in Function 0, and must control the Upstream Port's Link behavior on behalf of all the Functions of the device.

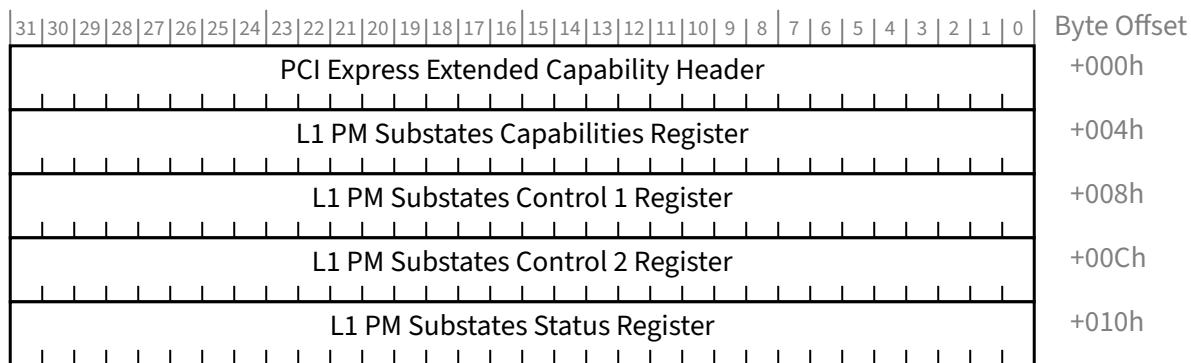


Figure 7-147 L1 PM Substates Extended Capability §

7.8.3.1 L1 PM Substates Extended Capability Header (Offset 00h) §

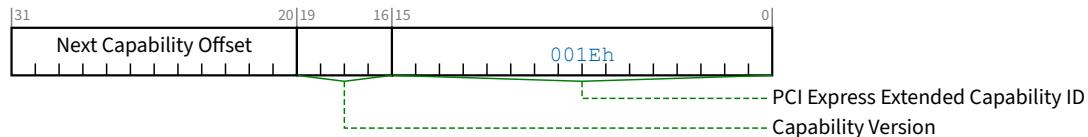


Figure 7-148 L1 PM Substates Extended Capability Header §

Table 7-131 L1 PM Substates Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for L1 PM Substates is 001Eh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. This field must be 2h if the L1 PM Substates Status Register is implemented and must be 1h otherwise. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits. | RO |

7.8.3.2 L1 PM Substates Capabilities Register (Offset 04h) §

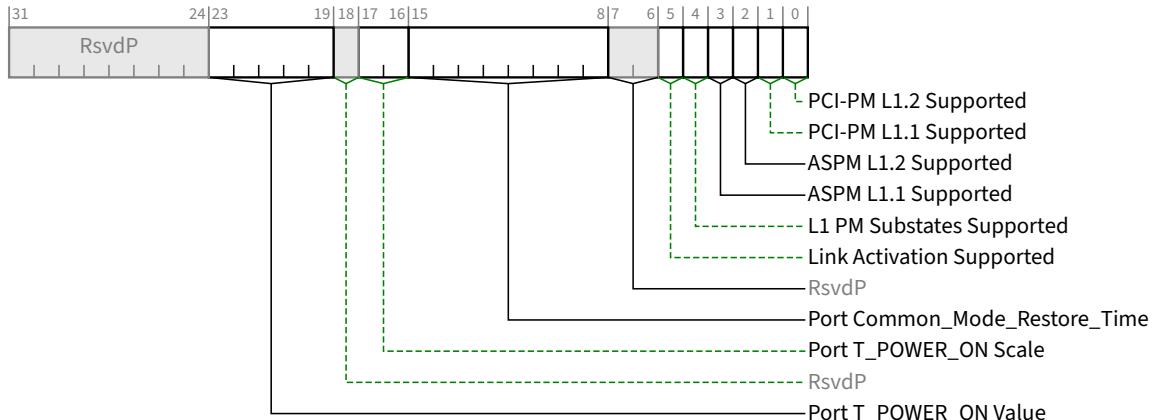


Figure 7-149 L1 PM Substates Capabilities Register §

Table 7-132 L1 PM Substates Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------------------|
| 0 | PCI-PM L1.2 Supported - When Set this bit indicates that PCI-PM <u>L1.2</u> is supported. | HwInit |
| 1 | PCI-PM L1.1 Supported - When Set this bit indicates that PCI-PM <u>L1.1</u> is supported, and must be Set by all Ports implementing L1 PM Substates . | HwInit |
| 2 | ASPM L1.2 Supported - When Set this bit indicates that ASPM <u>L1.2</u> is supported. | HwInit |
| 3 | ASPM L1.1 Supported - When Set this bit indicates that ASPM <u>L1.1</u> is supported. | HwInit |
| 4 | L1 PM Substates Supported - When Set this bit indicates that this Port supports <u>L1 PM Substates</u> . | HwInit |
| 5 | Link Activation Supported - For Downstream Ports, when Set, this bit indicates that this Port supports <u>Link Activation</u> . See § Section 5.5.6 for details. This bit is of type RsvdP for Upstream Ports. | HwInit / RsvdP |
| 15:8 | Port Common_Mode_Restore_Time - Time (in μ s) required for this Port to re-establish common mode as described in § Table 5-11 . Required for all Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP . | HwInit / RsvdP (See description) |
| 17:16 | Port T_POWER_ON Scale - Specifies the scale used for the <u>Port T_POWER_ON Value</u> field in the <u>L1 PM Substates Capabilities Register</u> . Range of Values 00b 2 μ s 01b 10 μ s 10b 100 μ s 11b Reserved | HwInit / RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------|
| | <p>Required for all Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP.</p> <p>Default value is 00b</p> | |
| 23:19 | <p>Port T_POWER_ON Value - Along with the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities Register sets the time (in μs) that this Port requires the port on the opposite side of Link to wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface.</p> <p>The value of Port T_POWER_ON is calculated by multiplying the value in this field by the scale value in the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities Register .</p> <p>Default value is 0 0101b</p> <p>Required for all Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP.</p> | HwInit / RsvdP |

7.8.3.3 L1 PM Substates Control 1 Register (Offset 08h) §

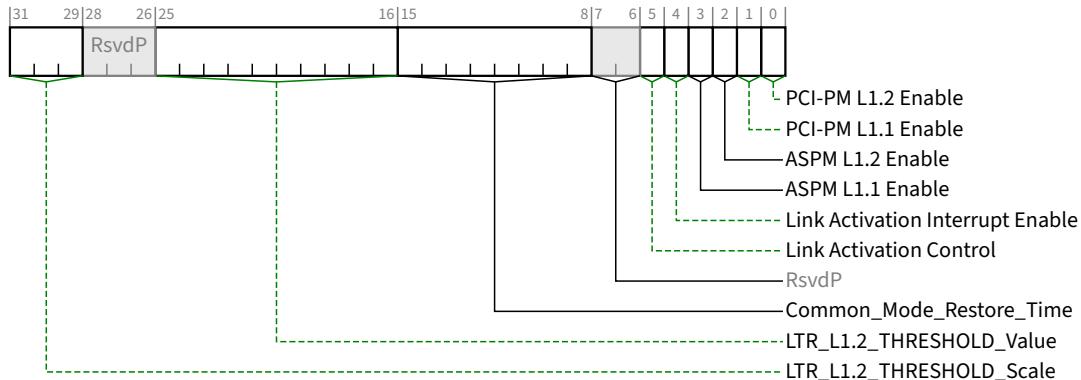


Figure 7-150 L1 PM Substates Control 1 Register §

Table 7-133 L1 PM Substates Control 1 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>PCI-PM L1.2 Enable - When Set this bit enables PCI-PM L1.2.</p> <p>Required for both Upstream and Downstream Ports. For Ports for which the PCI-PM L1.2 Supported bit is Clear this bit is permitted to be hardwired to 0.</p> <p>For compatibility with possible future extensions, software must not enable L1 PM Substates unless the L1 PM Substates Supported bit in the L1 PM Substates Capabilities Register is Set.</p> <p>Default value is 0b.</p> | RW |
| 1 | <p>PCI-PM L1.1 Enable - When Set this bit enables PCI-PM L1.1.</p> <p>Required for both Upstream and Downstream Ports.</p> <p>For compatibility with possible future extensions, software must not enable L1 PM Substates unless the L1 PM Substates Supported bit in the L1 PM Substates Capabilities Register is Set.</p> <p>Default value is 0b.</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|---------------------------------|
| 2 | <p>ASPM L1.2 Enable - When Set this bit enables ASPM L1.2.</p> <p>Required for both Upstream and Downstream Ports.</p> <p>For Ports for which the <u>ASPM L1.2 Supported</u> bit is Clear this bit is permitted to be hardwired to 0.</p> <p>For compatibility with possible future extensions, software must not enable <u>L1 PM Substates</u> unless the <u>L1 PM Substates Supported</u> bit in the <u>L1 PM Substates Capabilities Register</u> is Set.</p> <p>Default value is 0b.</p> | RW |
| 3 | <p>ASPM L1.1 Enable - When Set this bit enables ASPM L1.1.</p> <p>Required for both Upstream and Downstream Ports.</p> <p>For Ports for which the <u>ASPM L1.1 Supported</u> bit is Clear this bit is permitted to be hardwired to 0.</p> <p>For compatibility with possible future extensions, software must not enable <u>L1 PM Substates</u> unless the <u>L1 PM Substates Supported</u> bit in the <u>L1 PM Substates Capabilities Register</u> is Set.</p> <p>Default value is 0b.</p> | RW |
| 4 | <p>Link Activation Interrupt Enable - When set this bit enables the generation of an interrupt to indicate the completion of the <u>Link Activation</u> process. See § Section 5.5.6 for details.</p> <p>Required for Downstream Ports when the <u>Link Activation Supported</u> bit is Set, otherwise it is permitted to be hardwired to 0b.</p> <p>Must be <u>RsvdP</u> for Upstream Ports.</p> <p>Default value is 0b.</p> | RW / RsvdP |
| 5 | <p>Link Activation Control - When this bit is Set, the Port must initiate the <u>Link Activation</u> process. See § Section 5.5.6 for details.</p> <p>Required for Downstream Ports when the <u>Link Activation Supported</u> bit is Set, otherwise it is permitted to be hardwired to 0b.</p> <p>Must be <u>RsvdP</u> for Upstream Ports.</p> <p>Default value is 0b.</p> | RW / RsvdP |
| 15:8 | <p>Common Mode Restore Time - Sets value of $T_{COMMONMODE}$ (in μs), which must be used by the Downstream Port for timing the re-establishment of common mode, as described in § Table 5-11.</p> <p>This field must only be modified when the <u>ASPM L1.2 Enable</u> and <u>PCI-PM L1.2 Enable</u> bits are both Clear. The Port behavior is undefined if this field is modified when either the <u>ASPM L1.2 Enable</u> and/or <u>PCI-PM L1.2 Enable</u> bit(s) are Set.</p> <p>Required for Downstream Ports for which either the <u>PCI-PM L1.2 Supported</u> bit is Set, <u>ASPM L1.2 Supported</u> bit is Set, or both are Set, otherwise this field is of type <u>RsvdP</u>.</p> <p>This field is of type <u>RsvdP</u> for Upstream Ports.</p> <p>Default value is implementation specific.</p> | RW / RsvdP (See Description) |
| 25:16 | <p>LTR_L1.2_THRESHOLD_Value - Along with the <u>LTR_L1.2_THRESHOLD_Scale</u>, this field indicates the LTR threshold used to determine if entry into <u>L1</u> results in <u>L1.1</u> (if enabled) or <u>L1.2</u> (if enabled).</p> <p>The default value for this field is 00 0000 0000b.</p> <p>This field must only be modified when the <u>ASPM L1.2 Enable</u> bit is Clear. The Port behavior is undefined if this field is modified when the <u>ASPM L1.2 Enable</u> bit is Set.</p> <p>Required for all Ports for which the <u>ASPM L1.2 Supported</u> bit is Set, otherwise this field is of type <u>RsvdP</u>.</p> | RW / RsvdP (See Description) |
| 31:29 | <p>LTR_L1.2_THRESHOLD_Scale - This field provides a scale for the value contained within the <u>LTR_L1.2_THRESHOLD_Value</u>. Encoding is the same as the <u>LatencyScale</u> fields in the <u>LTR Message</u> (see § Section 6.18).</p> | RW / RsvdP (See description) |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>The default value for this field is 000b.</p> <p>Hardware operation is undefined if software writes a Not-Permitted value to this field.</p> <p>This field must only be modified when the ASPM L1.2 Enable bit is Clear. The Port behavior is undefined if this field is modified when the <u>ASPM L1.2 Enable</u> bit is Set.</p> <p>Required for all Ports for which the <u>ASPM L1.2 Supported</u> bit is Set, otherwise this field is of type RsvdP .</p> | |

7.8.3.4 L1 PM Substates Control 2 Register (Offset 0Ch) §

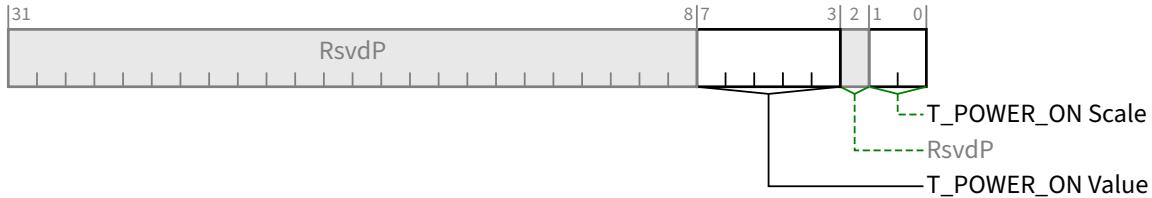


Figure 7-151 L1 PM Substates Control 2 Register §

Table 7-134 L1 PM Substates Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|---------------|
| 1:0 | <p>T_POWER_ON Scale - Specifies the scale used for <u>T_POWER_ON Value</u> .</p> <p>Range of Values:</p> <ul style="list-style-type: none"> 00b 2 μs 01b 10 μs 10b 100 μs 11b Reserved <p>Required for all Ports that support L1.2 , otherwise this field is of type RsvdP .</p> <p>This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. The Port behavior is undefined if this field is modified when either the <u>ASPM L1.2 Enable</u> and/or <u>PCI-PM L1.2 Enable</u> bit(s) are Set.</p> <p>Default value is 00b</p> | RW / RsvdP |
| 7:3 | <p>T_POWER_ON Value - Along with the <u>T_POWER_ON Scale</u> sets the minimum amount of time (in μs) that the Port must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface.</p> <p>T_POWER_ON is calculated by multiplying the value in this field by the value in the <u>T_POWER_ON Scale</u> field.</p> <p>This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. The Port behavior is undefined if this field is modified when either the <u>ASPM L1.2 Enable</u> and/or <u>PCI-PM L1.2 Enable</u> bit(s) are Set.</p> <p>Default value is 0 0101b</p> <p>Required for all Ports that support L1.2 , otherwise this field is of type RsvdP .</p> | RW / RsvdP |

7.8.3.5 L1 PM Substates Status Register (Offset 10h) §

Hardware must implement this register if the Capability Version in the L1 PM Substates Extended Capability Header is 2h or greater. This register is not present if the Capability Version is 1h.

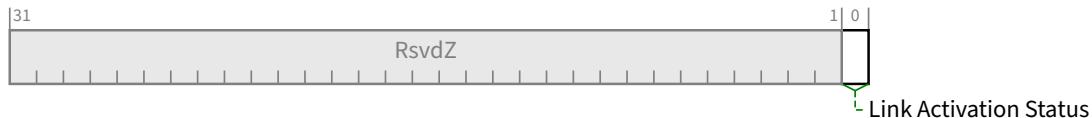


Figure 7-152 L1 PM Substates Status Register §

Table 7-135 L1 PM Substates Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|--------------|
| 0 | <p>Link Activation Status - Indicates the status of Link Activation . See § Section 5.5.6 for details.</p> <p>Required for Downstream Ports when the Link Activation Supported bit is Set, otherwise it is hardwired to 0b.</p> <p>Must be RsvdZ for Upstream Ports.</p> <p>Default value is 0b.</p> | RW1C / RsvdZ |

7.8.4 Advanced Error Reporting Extended Capability §

The PCI Express Advanced Error Reporting (AER) Capability is an optional Extended Capability that may be implemented by PCI Express device Functions supporting advanced error control and reporting. The Advanced Error Reporting Extended Capability structure definition has additional interpretation for Root Ports and Root Complex Event Collectors. Software must interpret the Device/Port Type field in the PCI Express Capabilities register to determine the availability of additional registers for Root Ports, RPs, and RCECs. RPs or RCECs that support Root Complex Event Collectors can distinguish ERR COR Messages targeting SFW vs the OS and handle them appropriately.

ECN: Base 6.3
RC-ECS△<1>

In an SR-IOV device, if AER is not implemented in a PF, it must not be implemented in its associated VFs. If AER is implemented in the PF, it is optional in its VFs.

In an SR-IOV device, the Header Log space for a PF is independent of any for its associated VFs and must be implemented with dedicated storage space. VFs that implement AER may share Header Log space among VFs associated with a single PF. Shared Header Log space must have storage for at least one header. See § Section 6.2.4.2.1 for further details.

§ Figure 7-153 and § Figure 7-154 show the PCI Express Advanced Error Reporting Extended Capability structure. In § Figure 7-154 , the last 6 DW are optional. Implementations are permitted to implement between 0 and 6 additional DW of Header Log (see Header Log Size for details).

Note that if an error reporting bit field is marked as optional in the error registers, the bits must be implemented or not implemented as a group across the Status, Mask and Severity registers. In other words, a Function is required to

implement the same error bit fields in corresponding Status, Mask and Severity registers. Bits corresponding to bit fields that are not implemented must be hardwired to 0, unless otherwise specified.

Base 6.4 vs Base 6.3

| | Byte Offset |
|---|-------------|
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | |
| PCI Express Extended Capability Header | +000h |
| Uncorrectable Error Status Register | +004h |
| Uncorrectable Error Mask Register | +008h |
| Uncorrectable Error Severity Register | +00Ch |
| Correctable Error Status Register | +010h |
| Correctable Error Mask Register | +014h |
| Advanced Error Capabilities and Control Register | +018h |
| Header Log Register | +01Ch |
| Root Error Command Register | +020h |
| Root Error Status Register | +024h |
| Error Source Identification Register | +028h |
| TLP Prefix Log Register | +02Ch |
| | +030h |
| | +034h |
| | +038h |
| | +03Ch |
| | +040h |
| | +044h |

Figure 7-153 Advanced Error Reporting Extended Capability - Functions that do not support Flit Mode Structure §

Base 6.4 vs Base 6.3

| | Byte Offset |
|---|-------------|
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | +000h |
| PCI Express Extended Capability Header | |
| Uncorrectable Error Status Register | +004h |
| Uncorrectable Error Mask Register | +008h |
| Uncorrectable Error Severity Register | +00Ch |
| Correctable Error Status Register | +010h |
| Correctable Error Mask Register | +014h |
| Advanced Error Capabilities and Control Register | +018h |
| Header Log Register DW1-4 | +01Ch |
| | +020h |
| | +024h |
| | +028h |
| Root Error Command Register | +02Ch |
| Root Error Status Register | +030h |
| Error Source Identification Register | +034h |
| | +038h |
| | +03Ch |
| | +040h |
| | +044h |
| | +048h |
| Header Log Register DW5-14 (text defines implementation requirements) | +04Ch |
| | +050h |
| | +054h |
| | +058h |
| | +05Ch |

Figure 7-154 Advanced Error Reporting Extended Capability - Functions that support Flit Mode Structure §

7.8.4.1 Advanced Error Reporting Extended Capability Header (Offset 00h) §

§ Figure 7-155 details the allocation of register fields of an Advanced Error Reporting Extended Capability header; § Table 7-136 provides the respective bit definitions.

Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Advanced Error Reporting Extended Capability is 0001h .

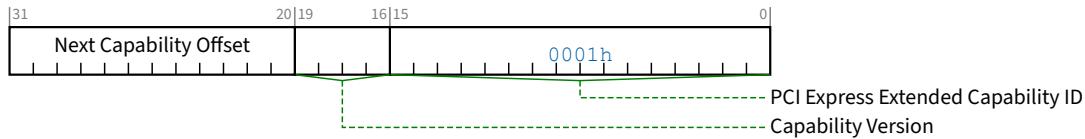


Figure 7-155 Advanced Error Reporting Extended Capability Header §

Table 7-136 Advanced Error Reporting Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Advanced Error Reporting Extended Capability is 0001h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. This field must be 3h if Flit Mode Supported is Set. This field must be 2h or 3h if End-End TLP Prefix Supported is Set (see § Section 7.5.3.15). Otherwise this field must be 1h, 2h, or 3h. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.8.4.2 Uncorrectable Error Status Register (Offset 04h) §

The Uncorrectable Error Status Register indicates error detection status of individual errors on a PCI Express device Function. An individual error status bit that is Set indicates that a particular error was detected; software may clear an error status by writing a 1b to the respective bit. Refer to § Section 6.2 for further details. Register bits not implemented by the Function are hardwired to 0b. § Figure 7-156 details the allocation of register fields of the Uncorrectable Error Status Register ; § Section 7.8.4.2 provides the respective bit definitions.

For ~~↑↓SR-IOV devices, ↑↓SR-IOV Devices, ↑~~ applicable errors categorized as non-Function-specific must be logged in PFs and non-IOV Functions, but not logged in VFs. VFs must log only Function-specific errors, so for VFs the intended attribute for applicable non-Function-specific errors is VF ROZ . This is mandatory for some, but is only strongly recommended for others since earlier versions of this specification required them to be RW1CS . Certain other errors are applicable only for Root Port, Switch Port, or Function 0 Endpoint Functions, so for VFs their intended attribute is VF ROZ , and they have the same issue with earlier versions of this specification. .

Base 6.4 vs Base 6.3

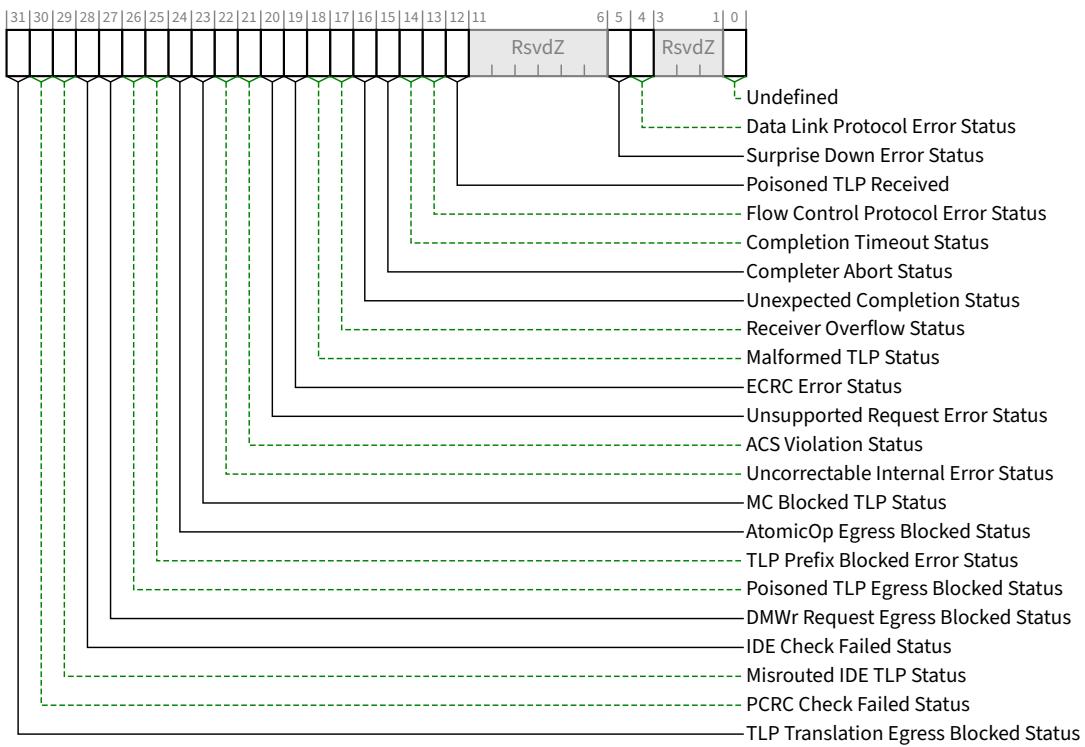


Figure 7-156 Uncorrectable Error Status Register §

Table 7-137 Uncorrectable Error Status Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|---|-----------------|-----------|
| 0 | Undefined Undefined - The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit. | Undefined | Undefined |
| 4 | Data Link Protocol Error Status | RW1CS VF ROZ | 0b |
| 5 | Surprise Down Error Status (Optional – Note 1) | RW1CS VF ROZ | 0b |
| 12 | Poisoned TLP Received Status | RW1CS | 0b |
| 13 | Flow Control Protocol Error Status (Optional – Note 1) | RW1CS VF ROZ | 0b |
| 14 | Completion Timeout Status ¹⁹⁴ | RW1CS | 0b |
| 15 | Completer Abort Status (Optional – Note 1) | RW1CS | 0b |

194. For Switch Ports, required if the Switch Port issues Non-Posted Requests [↑↑or UIO Requests↑↑](#) on its own behalf (vs. only forwarding such Requests generated by other devices). If the Switch Port does not issue such Requests, then the Completion Timeout mechanism is not applicable and this bit must be hardwired to 0b.

Errata: Base 6.3 B834Δ

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | Default |
|--------------|--|--|---------|
| 16 | Unexpected Completion Status | <u>RW1CS</u> | 0b |
| 17 | Receiver Overflow Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 18 | Malformed TLP Status | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 19 | ECRC Error Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 20 | Unsupported Request Error Status | <u>RW1CS</u> | 0b |
| 21 | ACS Violation Status (Optional – Note 1) | <u>RW1CS</u> | 0b |
| 22 | Uncorrectable Internal Error Status (Optional) | <u>RW1CS</u> | 0b |
| 23 | MC Blocked TLP Status (Optional – Note 1) | <u>RW1CS</u> | 0b |
| 24 | AtomicOp Egress Blocked Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 25 | TLP Prefix Blocked Error Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 26 | Poisoned TLP Egress Blocked Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 27 | DMWr Request Egress Blocked Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 28 | IDE Check Failed Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 29 | Misrouted IDE TLP Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 30 | PCRC Check Failed Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |
| 31 | TLP Translation Egress Blocked Status (Optional – Note 1) | <u>RW1CS</u> <u>VF ROZ</u> <u>(Note 2)</u> | 0b |

Notes:

1. Must implement if the corresponding Uncorrectable Error Mask bit is implemented. Otherwise, hardwire to 0.

| Bit Location | Register Description | Attributes | Default |
|---|----------------------|------------|---------|
| 2. VF ROZ is strongly recommended for VF Functions, but for backward compatibility with previous versions of this specification, RW1CS is permitted if the VF implements the corresponding Uncorrectable Error Mask Register bit. | | | |

7.8.4.3 Uncorrectable Error Mask Register (Offset 08h) §

The Uncorrectable Error Mask Register controls reporting of individual errors by the device Function to the PCI Express Root Complex via a PCI Express error Message. A masked error (respective bit Set in the mask register) is not recorded or reported in the Header Log, TLP Prefix Log, or First Error Pointer, and is not reported to the PCI Express Root Complex by this Function. Refer to § Section 6.2 for further details. There is a mask bit per error bit of the Uncorrectable Error Status register. Register fields for bits not implemented by the Function are hardwired to 0b. § Figure 7-157 details the allocation of register fields of the Uncorrectable Error Mask Register; § Table 7-138 provides the respective bit definitions.

For VF fields marked as VF RsvdP, the associated PF's setting applies to the VF. For VF fields marked as VF ROZ, the error is not applicable to a VF.

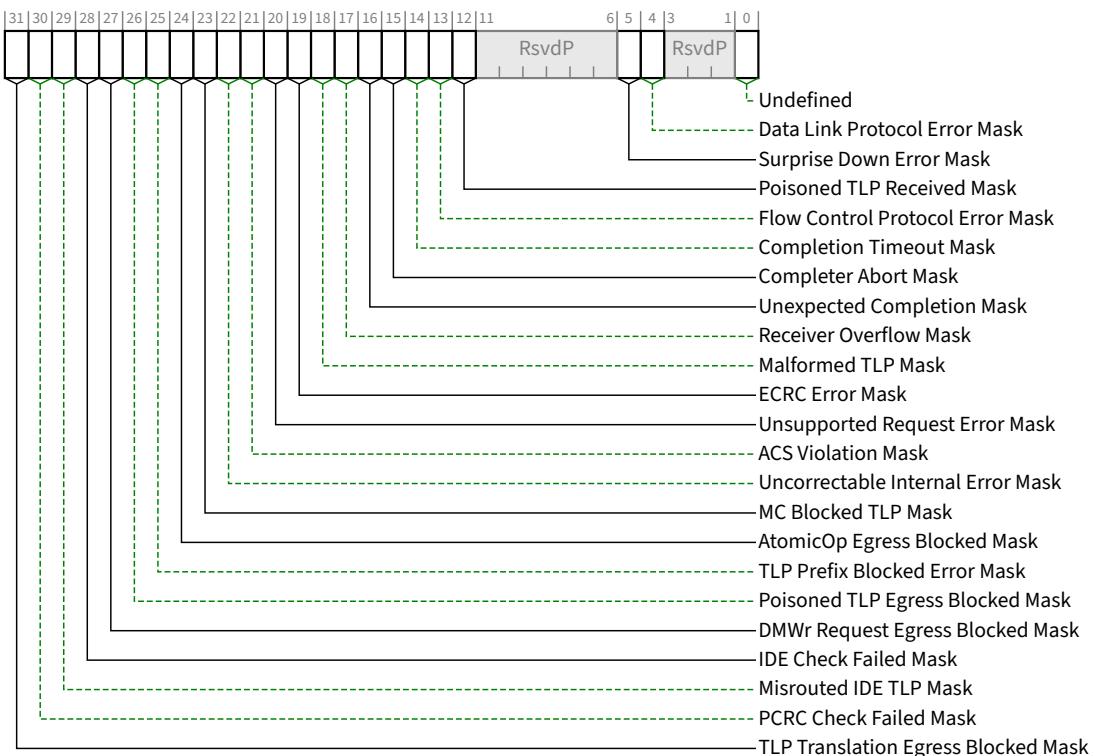


Figure 7-157 Uncorrectable Error Mask Register §

Base 6.4 vs Base 6.3

Table 7-138 Uncorrectable Error Mask Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|---|-------------------------|-----------|
| 0 | Undefined Undefined - The value read from this bit is undefined. In previous versions of this specification, this bit was used to mask a Link Training Error. System software must ignore the value read from this bit. System software must only write a value of 1b to this bit. | Undefined | Undefined |
| 4 | Data Link Protocol Error Mask | RWS VF ROZ | 0b |
| 5 | Surprise Down Error Mask (Surprise Down Error Reporting Capable – Note 1) | RWS VF ROZ | 0b |
| 12 | Poisoned TLP Received Mask | RWS VF RsvdP | 0b |
| 13 | Flow Control Protocol Error Mask (Optional) | RWS VF ROZ | 0b |
| 14 | Completion Timeout Mask ¹⁹⁵ | RWS VF RsvdP | 0b |
| 15 | Completer Abort Mask (Optional) | RWS VF RsvdP | 0b |
| 16 | Unexpected Completion Mask | RWS VF RsvdP | 0b |
| 17 | Receiver Overflow Mask (Optional) | RWS VF ROZ | 0b |
| 18 | Malformed TLP Mask | RWS VF ROZ | 0b |
| 19 | ECRC Error Mask (ECRC Check Capable – Note 1) | RWS VF ROZ | 0b |
| 20 | Unsupported Request Error Mask | RWS VF RsvdP | 0b |
| 21 | ACS Violation Mask (ACS Extended Capability – Note 2) | RWS VF RsvdP | 0b |
| 22 | Uncorrectable Internal Error Mask (Optional) | RWS | 1b |
| 23 | MC Blocked TLP Mask (Multicast Extended Capability – Note 2) | RWS | 0b |
| 24 | AtomicOp Egress Blocked Mask (AtomicOp Egress Blocking – Note 3) | RWS VF ROZ Note 6 | 0b |

195. For Switch Ports, required if the Switch Port issues Non-Posted Requests [↑↑or UIO Requests↑↑](#) on its own behalf (vs. only forwarding such Requests generated by other devices). If the Switch Port does not issue such Requests, then the Completion Timeout mechanism is not applicable and this bit must be hardwired to 0b.

Errata: Base 6.3 B834Δ

| Bit Location | Register Description | Attributes | Default |
|--------------|---|-------------------------|---------|
| 25 | TLP Prefix Blocked Error Mask (<u>End-End TLP Prefix Supported – Note 3</u> , <u>OHC-E Support – Note 8</u>) | RWS VF ROZ Note 6 | 0b |
| 26 | Poisoned TLP Egress Blocked Mask (<u>Poisoned TLP Egress Blocking Supported – Note 3</u>) | RWS VF ROZ Note 6 | 1b |
| 27 | DMWr Request Egress Blocked Mask (<u>DMWr Egress Blocking – Note 3</u>) | RWS VF ROZ Note 6 | 0b |
| 28 | IDE Check Failed Mask (<u>IDE Extended Capability – Note 4</u>) | RWS VF ROZ Note 6 | 0b |
| 29 | Misrouted IDE TLP Mask (<u>IDE Extended Capability – Note 4</u>) | RWS VF ROZ Note 6 | 0b |
| 30 | PCRC Check Failed Mask (<u>PCRC Supported – Note 5</u>) | RWS VF ROZ Note 6 | 0b |
| 31 | TLP Translation Egress Blocked Mask (Routing elements that translate between FM and NFM in either direction must implement this bit (see § <u>Section 2.2.1.2</u>)) | RWS VF ROZ Note 6 | 0b |

Notes:

1. When Set in a non-VF Function, this AER mask bit must be implemented. Otherwise, for non-VF Functions, it is optional.
2. When a Function implements this Extended Capability (i.e., ACS/Multicast), that Function must implement this AER mask bit.
3. When Set in a non-VF Function, that Function must implement this AER mask bit.
4. When Function 0 implements the IDE Extended Capability, and if Function 0 implements AER, then Function 0 must implement this AER mask bit.
5. When PCRC Supported is Set in Function 0, and if Function 0 implements AER, then Function 0 must implement this AER mask bit.
6. VF ROZ is strongly recommended for VF Functions, but for backward compatibility with previous versions of this specification, RWS is permitted.
7. Placeholder
8. For Root Ports and Switch Ports, where the associated OHC-E Support field is 111b, 001b, 010b, 011b or 100b, that Port must implement this AER mask bit.

7.8.4.4 Uncorrectable Error Severity Register (Offset 0Ch) §

The Uncorrectable Error Severity Register controls whether an individual error is reported as a Non-fatal or Fatal error. An error is reported as fatal when the corresponding error bit in the severity register is Set. If the bit is Clear, the corresponding error is considered non-fatal. Refer to § Section 6.2 for further details. Register fields for bits not

implemented by the Function are hardwired to an implementation specific value. § Figure 7-158 details the allocation of register fields of the Uncorrectable Error Severity Register ; § Table 7-139 provides the respective bit definitions.

For VF fields marked as VF RsvdP , the associated PF's setting applies to the VF. For VF fields marked as VF ROZ , the error is not applicable to a VF.

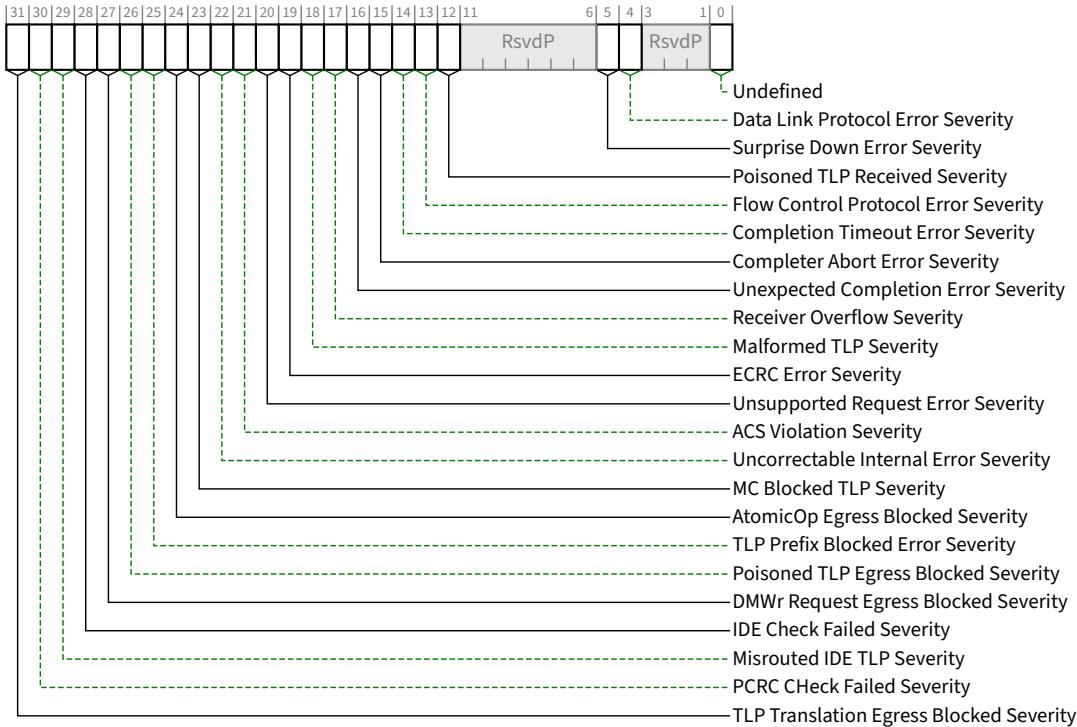


Figure 7-158 Uncorrectable Error Severity Register §

Table 7-139 Uncorrectable Error Severity Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|-----------------|-----------|
| 0 | Undefined Undefined - The value read from this bit is undefined. In previous versions of this specification, this bit was used to Set the severity of a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit. | Undefined | Undefined |
| 4 | Data Link Protocol Error Severity | RWS VF ROZ | 1b |
| 5 | Surprise Down Error Severity (Optional – Note 1) | RWS VF ROZ | 1b |
| 12 | Poisoned TLP Received Severity | RWS VF RsvdP | 0b |
| 13 | Flow Control Protocol Error Severity (Optional – Note 1) | RWS VF ROZ | 1b |
| 14 | Completion Timeout Error Severity ¹⁹⁶ | RWS | 0b |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | Default |
|--------------|--|-------------------------|---------|
| | | VF RsvdP | |
| 15 | Completer Abort Error Severity (Optional – Note 1) | RWS VF RsvdP | 0b |
| 16 | Unexpected Completion Error Severity | RWS VF RsvdP | 0b |
| 17 | Receiver Overflow Severity (Optional – Note 1) | RWS VF ROZ | 1b |
| 18 | Malformed TLP Severity | RWS VF ROZ | 1b |
| 19 | ECRC Error Severity (Optional – Note 1) | RWS VF ROZ | 0b |
| 20 | Unsupported Request Error Severity | RWS VF RsvdP | 0b |
| 21 | ACS Violation Severity (Optional – Note 1) | RWS VF RsvdP | 0b |
| 22 | Uncorrectable Internal Error Severity (Optional – Note 1) | RWS | 1b |
| 23 | MC Blocked TLP Severity (Optional – Note 1) | RWS | 0b |
| 24 | AtomicOp Egress Blocked Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 0b |
| 25 | TLP Prefix Blocked Error Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 0b |
| 26 | Poisoned TLP Egress Blocked Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 0b |
| 27 | DMWr Request Egress Blocked Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 0b |
| 28 | IDE Check Failed Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 1b |
| 29 | Misrouted IDE TLP Severity (Optional – Note 1) | RWS | 0b |

196. For Switch Ports, required if the Switch Port issues Non-Posted Requests [↑↑or UIO Requests↑↑](#) on its own behalf (vs. only forwarding such Requests generated by other devices). If the Switch Port does not issue such Requests, then the Completion Timeout mechanism is not applicable and this bit must be hardwired to 0b.

Errata: Base 6.3 B834Δ

| Bit Location | Register Description | Attributes | Default |
|--------------|--|-------------------------|---------|
| | | VF ROZ Note 2 | |
| 30 | PCRC Check Failed Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 0b |
| 31 | TLP Translation Egress Blocked Severity (Optional – Note 1) | RWS VF ROZ Note 2 | 0b |

Notes:

1. Must implement if the corresponding Uncorrectable Error Mask Register bit is implemented. Otherwise, hardwire to 0.
2. VF ROZ is strongly recommended for VF Functions, but for backward compatibility with previous versions of this specification, RWS is permitted if the VF implements the corresponding Uncorrectable Error Mask Register bit.

7.8.4.5 Correctable Error Status Register (Offset 10h) §

The Correctable Error Status register reports error status of individual correctable error sources on a PCI Express device Function. When an individual error status bit is Set, it indicates that a particular error occurred; software may clear an error status by writing a 1b to the respective bit. Refer to § Section 6.2 for further details. Register bits not implemented by the Function are hardwired to 0b. § Figure 7-159 details the allocation of register fields of the Correctable Error Status register; § Table 7-140 provides the respective bit definitions.

For ~~↑↓SR-IOV devices, ↑↓SR-IOV Devices, ↑~~ errors categorized as non-Function-specific must be logged in PFs and non-IOV Functions, but not logged in VFs. VFs must log only Function-specific errors.

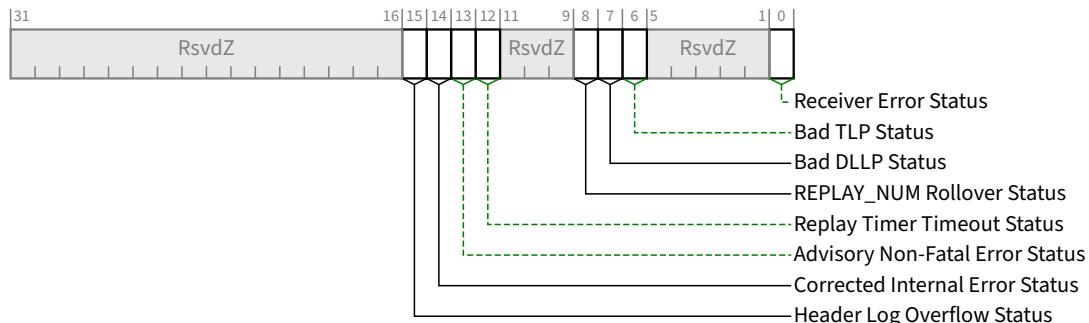


Figure 7-159 Correctable Error Status Register §

Table 7-140 Correctable Error Status Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|---|-----------------|---------|
| 0 | Receiver Error Status ¹⁹⁷ | RW1CS VF ROZ | 0b |

| Bit Location | Register Description | Attributes | Default |
|--------------|---|-------------------------------|---------|
| 6 | Bad TLP Status | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 7 | Bad DLLP Status | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 8 | REPLAY_NUM Rollover Status | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 12 | Replay Timer Timeout Status | <u>RW1CS</u> <u>VF ROZ</u> | 0b |
| 13 | Advisory Non-Fatal Error Status | <u>RW1CS</u> | 0b |
| 14 | Corrected Internal Error Status (Optional) | <u>RW1CS</u> | 0b |
| 15 | Header Log Overflow Status (Optional) If the VF implements Header Log sharing (see § Section 6.2.4.2.1), this bit must be hardwired to Zero. | <u>RW1CS / 0b</u> | 0b |

7.8.4.6 Correctable Error Mask Register (Offset 14h) [§](#)

The Correctable Error Mask Register controls reporting of individual correctable errors by this Function to the PCI Express Root Complex via a PCI Express error Message. A masked error (respective bit Set in the mask register) is not reported to the PCI Express Root Complex by this Function. Refer to § [Section 6.2](#) for further details. There is a mask bit per error bit in the Correctable Error Status register. Register fields for bits not implemented by the Function are hardwired to 0b. § [Figure 7-160](#) details the allocation of register fields of the Correctable Error Mask Register ; § [Table 7-141](#) provides the respective bit definitions.

For VF fields marked as VF RsvdP , the associated PF's setting applies to the VF.

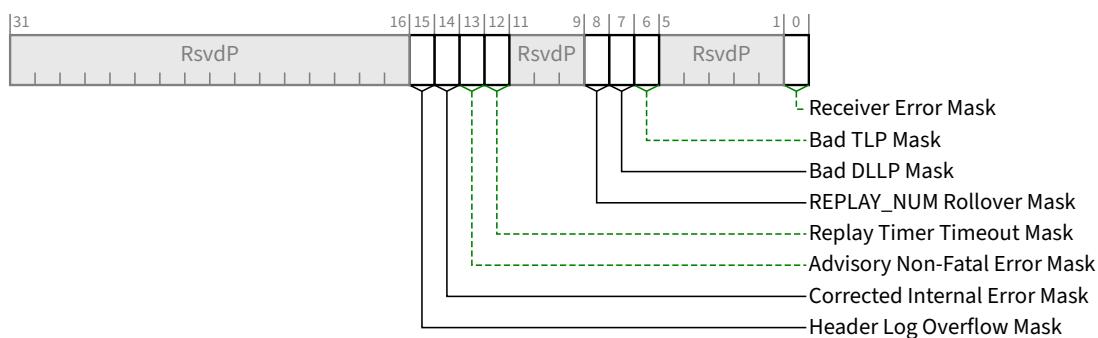


Figure 7-160 Correctable Error Mask Register [§](#)

197. For historical reasons, implementation of this bit is optional. If not implemented, this bit must be RsvdZ , and bit 0 of the Correctable Error Mask Register must also not be implemented. Note that some checking for Receiver Errors is required in all cases (see § [Section 4.2.1.1.3](#) , § [Section 4.2.5.8](#) , and § [Section 4.2.7](#)).

Table 7-141 Correctable Error Mask Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|---|-----------------|---------|
| 0 | Receiver Error Mask ¹⁹⁸ | RWS VF RsvdP | 0b |
| 6 | Bad TLP Mask | RWS VF RsvdP | 0b |
| 7 | Bad DLLP Mask | RWS VF RsvdP | 0b |
| 8 | REPLAY_NUM Rollover Mask | RWS VF RsvdP | 0b |
| 12 | Replay Timer Timeout Mask | RWS VF RsvdP | 0b |
| 13 | Advisory Non-Fatal Error Mask - This bit is Set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting . | RWS VF RsvdP | 1b |
| 14 | Corrected Internal Error Mask (Optional) | RWS | 1b |
| 15 | Header Log Overflow Mask (Optional) If the VF implements Header Log sharing (see § Section 6.2.4.2.1), this bit is RsvdP . | RWS / RsvdP | 1b |

7.8.4.7 Advanced Error Capabilities and Control Register (Offset 18h) §

§ Figure 7-161 details allocation of register fields in the Advanced Error Capabilities and Control register; § Table 7-142 provides the respective bit definitions. Handling of multiple errors is discussed in § Section 6.2.4.2 .

For VF fields marked as VF RsvdP , the associated PF's setting applies to the VF. For VF fields marked as VF ROZ , the error is not applicable to a VF.

198. For historical reasons, implementation of this bit is optional. If not implemented, this bit must be RsvdP , and bit 0 of the Correctable Error Status register must also not be implemented. Note that some checking for Receiver Errors is required in all cases (see § Section 4.2.1.1.3 , § Section 4.2.5.8 , and § Section 4.2.7).

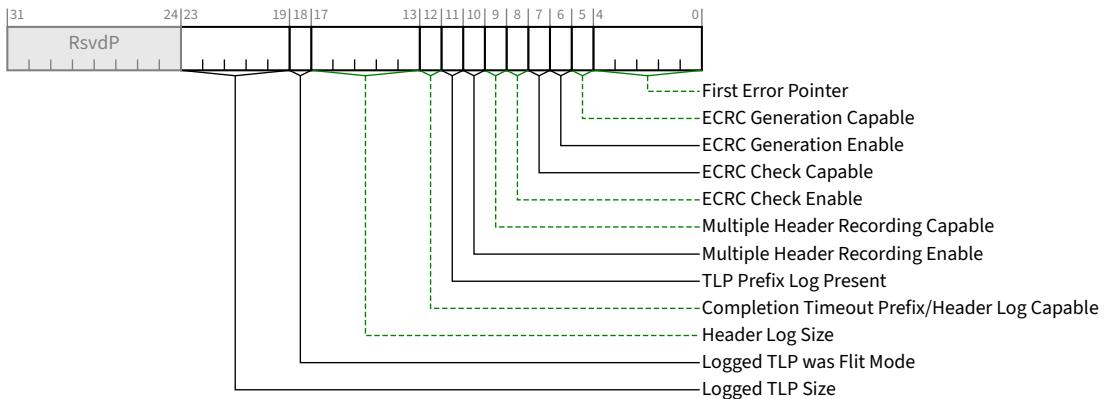


Figure 7-161 Advanced Error Capabilities and Control Register §

Table 7-142 Advanced Error Capabilities and Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------------------|
| 4:0 | First Error Pointer - The First Error Pointer is a field that identifies the bit position of the first error reported in the Uncorrectable Error Status register. Refer to § Section 6.2 for further details. | <u>ROS</u> |
| 5 | ECRC Generation Capable - If Set, this bit indicates that the Function is capable of generating ECRC (see § Section 2.7). | <u>RO</u> |
| 6 | ECRC Generation Enable - When Set, ECRC generation is enabled (see § Section 2.7). Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b. | <u>RWS</u> <u>VF RsvdP</u> |
| 7 | ECRC Check Capable - If Set, this bit indicates that the Function is capable of checking ECRC (see § Section 2.7). | <u>RO</u> |
| 8 | ECRC Check Enable - When Set, ECRC checking is enabled (see § Section 2.7). Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b. | <u>RWS</u> <u>VF RsvdP</u> |
| 9 | Multiple Header Recording Capable - If Set, this bit indicates that the Function is capable of recording more than one error header. Refer to § Section 6.2 for further details. If the VF implements Header Log sharing (see § Section 6.2.4.2.1), this bit must be hardwired to Zero. | <u>RO / 0b</u> |
| 10 | Multiple Header Recording Enable - When Set, this bit enables the Function to record more than one error header. Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. If the VF implements Header Log sharing (see § Section 6.2.4.2.1), this bit is <u>RsvdP</u> . Default value of this bit is 0b. | <u>RWS / RsvdP</u> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|-------------------|
| 11 | <p>TLP Prefix Log Present - If End-End TLP Prefix Supported is Clear, this bit is RsvdP.</p> <p>If Flit Mode Supported is Set, First Error Pointer is valid, and Logged TLP was Flit Mode is Set, this bit must be 0.</p> <p>If this bit is Set and First Error Pointer is valid, the TLP Prefix Log Register (offset 38h to 44h, also known as Header Log Register DW5-8) contains valid Non-Flit Mode End-End TLP Prefix information.</p> <p>If this bit is Clear or First Error Pointer not valid, the TLP Prefix Log Register does not contain End-End TLP Prefix information (the overlapping field, Header Log Register DW5-8, may contain Flit Mode TLP Header information as specified elsewhere in this section).</p> <p>Default value of this bit is 0.</p> <p>If the VF implements Header Log Sharing (see § Section 6.2.4.2.1), this bit must be Zero when the Header Log contains all 1s due to an overflow condition.</p> | ROS / RsvdP |
| 12 | <p>Completion Timeout Prefix/Header Log Capable - If Set, this bit indicates that the Function records the prefix/header of Request TLPs that experience a Completion Timeout error.</p> | HwInit |
| 17:13 | <p>Header Log Size - This field indicates the number of DW of Header Log that are implemented.</p> <p>If Flit Mode Supported is Set, see text for requirements.</p> <p>If Flit Mode Supported is Clear and End-End TLP Prefix Supported is Set, this value must either be 0 or must be greater than or equal to 8.</p> <p>If this field is 0 and Flit Mode Supported is Clear, the size of the header log depends on End-End TLP Prefix Supported . If End-End TLP Prefix Supported is Clear, the Header Log is 4 DW, otherwise the Header Log is 8 DW.</p> | HwInit / RsvdP |
| 18 | <p>Logged TLP was Flit Mode -- If Flit Mode Supported is Set, First Error Pointer is valid, and this bit is Set, the logged TLP was captured in Flit Mode otherwise the TLP was captured in Non-Flit Mode.</p> | ROS |
| 23:19 | <p>Logged TLP Size -- If Flit Mode Supported is Set and First Error Pointer is valid, this field contains the number of DW that were logged in the Header Log Register and, if appropriate, the TLP Prefix Log Register .</p> <p>If Flit Mode Supported is Set, First Error Pointer is valid, the VF implements Header Log Sharing (see § Section 6.2.4.2.1), and a Header Log overflow condition occurred, this field must be 0 (in addition to the Header Log containing all 1s).</p> | ROS |
| 1↑24↑ | <p>ECN: Base 5.3 RC-ECS[△]</p> <p>RC ECS Handling Capable - If Set, this bit indicates that the Function supports RC ECS Handling . This bit is only applicable to Root Ports and Root Complex Event Collectors; otherwise it is RsvdP.[↑]</p> | 1↑HwInit / RsvdP↑ |

7.8.4.8 Header Log Register (Offset 1Ch) §

The Header Log Register contains the header for the TLP corresponding to a detected error; refer to § Section 6.2 for further details. § Section 6.2 also describes the conditions where the packet header is recorded. This register is 16 bytes and adheres to the format of the headers defined throughout this specification.

The header is captured such that, when read using DW accesses, the fields of the header are laid out in the same way the headers are presented in this document. Therefore, byte 0 of the header is located in byte 3 of the Header Log Register , byte 1 of the header is in byte 2 of the Header Log Register and so forth. For 12-byte headers, only bytes 0 through 11 of the Header Log Register are used and values in bytes 12 through 15 are undefined.

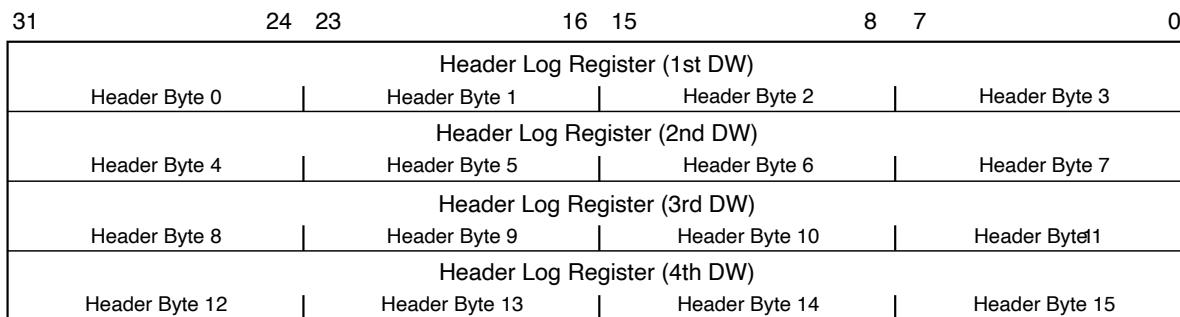
See § [Section 6.2.4.2.1](#) for further requirements when VFs share Header Log space.

In certain cases where a Malformed TLP is reported, the Header Log Register may contain TLP Prefix information. See § [Section 6.2.4.4](#) for details.

§ [Figure 7-162](#) details allocation of register fields in the Header Log Register ; § [Table 7-143](#) provides the respective bit definitions.

When Flit Mode Supported is Set and the link is operating in Flit Mode, the Header Log Register extends into additional DWs as indicated by the Header Log Size field. Software must parse the Type and OHC fields to determine the size and layout of a TLP recorded in the Header Log Register . Hardware is not required to support logging of TLP Headers larger than the largest size supported by the Port. Hardware is not required to support logging of OHC types not supported by the Port. TLP Trailers are not logged in the Header Log Register . The required minimum size of the Header Log Register is determined by the largest Header Base Size implemented by the Port (up to the maximum defined of 7 DW - See § [Table 2-5](#)), plus the largest number of OHC implemented by the Port (up to the maximum defined of 7 DW). Hardware must hardwire to zero the DW of the Header Log Register beyond those required to log the largest supported TLP Header and the overall length of the Advanced Error Reporting Extended Capability is reduced accordingly. As in Non-Flit Mode, Local TLP Prefixes are not logged.

When the link is operating in Non-Flit Mode, End-End TLP Prefixes are logged in the TLP Prefix Log Register .



OM14549A

[Figure 7-162 Header Log Register](#)

[Table 7-143 Header Log Register](#)

| Bit Location | Register Description | Attributes | Default |
|--------------|-------------------------------------|------------|---------|
| 127:0 | Header of TLP associated with error | ROS | 0 |

[7.8.4.9 Root Error Command Register \(Offset 2Ch\)](#)

The Root Error Command Register allows further control of Root Complex response to Correctable, Non-Fatal, and Fatal error Messages than the basic Root Complex capability to generate system errors in response to error Messages (either received or internally generated). Bit fields (see § [Figure 7-163](#)) enable or disable generation of interrupts (claimed by the Root Port or Root Complex Event Collector) in addition to system error Messages according to the definitions in § [Table 7-144](#) .

For both Root Ports and Root Complex Event Collectors , in order for a received error Message or an internally generated error Message to generate an interrupt enabled by this register, the error Message must be enabled for “transmission” by the Root Port or Root Complex Event Collector (see § [Section 6.2.4.1](#) and § [Section 6.2.8.1](#)).

For Functions other than Root Ports and Root Complex Event Collectors : when End-End TLP Prefix Supported is Set or Flit Mode Supported is Set, this register is RsvdP , otherwise Clear, this register is not required to be implemented.

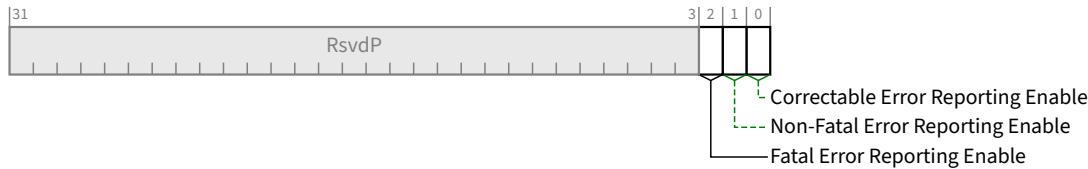


Figure 7-163 Root Error Command Register §

Table 7-144 Root Error Command Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|-----------------------------------|-----------------------|
| 0 | <p>Correctable Error Reporting Enable - When Set, this bit enables the generation of an interrupt ↑↑to the OS↑ when ↑↑a↑↑an applicable↑ correctable error is reported by any of the Functions in the Hierarchy Domain associated with this Root Port .</p> <p>Root Complex Event Collectors provide support for the above described functionality for RCIEPs .</p> <p>↑↑If RC ECS Handling Capable is Set, additional control bits determine which ERR_COR Messages are applicable for being handled via an interrupt to the OS versus being handled by SFW. See ECS SIG_SFW Handling by SFW Enable , ECS SIG_OS Handling by SFW Enable , and ECS Legacy Handling by SFW Enable .↑</p> <p>Refer to § Section 6.2 for further details.</p> | RW | 0b |
| 1 | <p>Non-Fatal Error Reporting Enable - When Set, this bit enables the generation of an interrupt when a Non-fatal error is reported by any of the Functions in the Hierarchy Domain associated with this Root Port .</p> <p>Root Complex Event Collectors provide support for the above described functionality for RCIEPs .</p> <p>Refer to § Section 6.2 for further details.</p> | RW | 0b |
| 2 | <p>Fatal Error Reporting Enable - When Set, this bit enables the generation of an interrupt when a Fatal error is reported by any of the Functions in the Hierarchy Domain associated with this Root Port .</p> <p>Root Complex Event Collectors provide support for the above described functionality for RCIEPs .</p> <p>Refer to § Section 6.2 for further details.</p> | RW | 0b |
| | <p>↑↑3↑ ECN: Base 6.3 RC-ECS△↔</p> <p>↑↑ECS Legacy Handling by SFW Enable – When Set, this bit enables signaling SFW (via a system-specific mechanism) when an ECS Legacy ERR COR Message is received. This handling is not contingent upon the Correctable Error Reporting Enable bit being Set, and no interrupt to the OS is generated.↑</p> | ↑↑RW / RsvdP↑ | ↑↑0b↑ |

| Bit Location | Register Description | Attributes | Default |
|--------------|--|---------------|---------|
| ↑↑4↑ | <p>↑↑This event is logged in the SFW Handling ERR_COR Received bit, the Multiple SFW Handling ERR_COR Received bit, and the SFW Handling Source ID field. This event is not logged in the ERR_COR Received bit, the Multiple ERR_COR Received bit, or the ERR_COR Source Identification field.↑</p> <p>↑↑If RC ECS Handling Capable is Clear, this bit is RsvdP.↑</p> <p>ECN: Base 6.3 RC-ECS△↔</p> | | |
| ↑↑5↑ | <p>↑↑ECS SIG_SFW Handling by SFW Enable – This bit has identical semantics to the ECS Legacy Handling by SFW Enable bit, with the exception that it applies to the ECS SIG_SFW subclass.↑</p> <p>↑↑If RC ECS Handling Capable is Clear, this bit is RsvdP.↑</p> <p>ECN: Base 6.3 RC-ECS△↔</p> | ↑↑RW / RsvdP↑ | ↑↑0b↑ |
| ↑↑31:16↑ | <p>ECN ↑↑ECS SIG_OS Handling by SFW Enable – This bit has identical semantics to the ECS Legacy Handling by SFW Enable bit, with the exception that it applies to the ECS SIG_OS subclass.↑</p> <p>↑↑If RC ECS Handling Capable is Clear, this bit is RsvdP.↑</p> <p>ECN: Base 6.3 RC-ECS△↔</p> | ↑↑RW / RsvdP↑ | ↑↑0b↑ |

IMPLEMENTATION NOTE: CONCURRENT MODIFICATIONS TO THE ROOT ERROR COMMAND REGISTER §

↑↑Both SFW and the OS may participate in the configuration the Root Error Command Register. It is envisioned that SFW will initially configure this register with platform-specific values, then if SFW and the OS negotiate ownership of the AER Extended Capability (e.g., via the _OSC mechanism defined in the PCI Firmware Specification), SFW may update this register during the negotiation process before returning control to the OS. Afterwards, if the OS owns the AER Extended Capability, the OS may update this register with OS-specific values. With this envisioned use case, SFW and the OS never update this register concurrently.↑

ECN: Base 6.3
RC-ECS△↔

↑↑If SFW and the OS ever update this register concurrently without proper coordination, one may unintentionally overwrite modifications performed by the other, due to each doing a read/modify/write sequence as required by RsvdP semantics. If this occurs, the results are undefined.↑

↑↑If an RP or RCEC supports RC ECS Handling, and the platform/OS supports use cases where both SFW and the OS handle events signaled by ERR_COR, there may be a need for both SFW and the OS to modify this register concurrently. The likelihood of this is envisioned to be small, since the configuration bits in this register are generally configured prior to runtime, and not modified during runtime. However, if a use case requires this, some mechanism outside the scope of this specification will need to coordinate these modification accesses. For example, SFW might define an interface for the OS to use when it needs to modify this register.↑

System error generation in response to PCI Express error Messages may be turned off by system software using the PCI Express Capability structure described in § Section 7.5.3 when advanced error reporting via interrupts is enabled. Refer to § Section 6.2 for further details.

7.8.4.10 Root Error Status Register (Offset 30h) §

The Root Error Status Register reports status of error Messages (↓↑(+) ↑↑(applicable) 199 ERR_COR , ERR_NONFATAL , and ERR_FATAL) received by the Root Port , and of errors detected by the Root Port itself (which are treated conceptually as if the Root Port had sent an error Message to itself). In order to update this register, error Messages received by the Root Port and/or internally generated error Messages must be enabled for “transmission” by the primary interface of the Root Port . ERR_NONFATAL and ERR_FATAL Messages are grouped together as uncorrectable. Each correctable and uncorrectable (Non-fatal and Fatal) error source has a first error bit and a next error bit associated with it respectively. When an error is received by a Root Complex, the respective first error bit is Set and the Requester ID is logged in the Error Source Identification Register . A Set individual error status bit indicates that a particular error category occurred; software may clear an error status by writing a 1b to the respective bit. If software does not clear the first reported error before another error Message is received of the same category (correctable or uncorrectable), the corresponding next error status bit will be set but the Requester ID of the subsequent error Message is discarded. The next error status bits may be cleared by software by writing a 1b to the respective bit as well. Refer to § Section 6.2 for further details. This register is updated regardless of the settings of the Root Control Register and the Root Error Command Register . § Figure 7-164 details allocation of register fields in the Root Error Status Register ; § Table 7-145 provides the respective bit definitions. Root Complex Event Collectors provide support for the above-described functionality for RCIEPs (and for the Root Complex Event Collector itself). In order to update this register, error Messages received by the Root Complex Event Collector from its associated RCIEPs and/or internally generated error Messages must be enabled for “transmission” by the Root Complex Event Collector .

ECN: Base 6.3
RC-ECS△
↓

For Functions other than Root Ports and Root Complex Event Collectors : when End-End TLP Prefix Supported is Set or Flit Mode Supported is Set, this register is RsvdZ , otherwise this register is not required to be implemented.

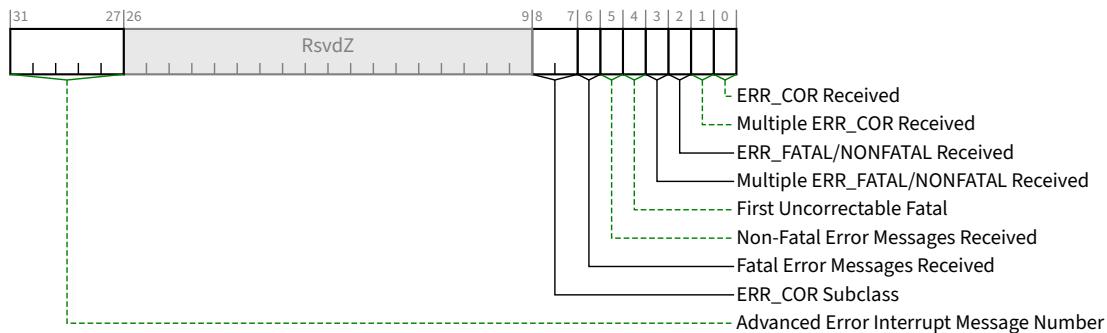


Figure 7-164 Root Error Status Register §

Table 7-145 Root Error Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|---|
| 0 | <p>ERR_COR Received - Set when ↓↑(+) ↑↑(applicable) Correctable error Message is received and this bit is not already Set.</p> <p>↓↑If RC ECS Handling Capable is Set, other control bits determine which ERR_COR Messages are applicable for logging by this bit. See § Section 6.2.4.1.3 .↑</p> <p>Default value of this bit is 0b.</p> | <p>ECN: Base 6.3 RC-ECS△ ↓</p> <p>RW1CS</p> |

199. ↑↑To see which ERR_COR Messages are not applicable for this description, see § Section 6.2.4.1.3 .↑

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|---|
| 1 | <p>Multiple ERR_COR Received - Set when If an applicable Correctable error Message is received and <u>ERR_COR Received</u> is already Set.</p> <p>If RC ECS Handling Capable is Set, other control bits determine which ERR COR Messages are applicable for logging by this bit. See § Section 6.2.4.1.3.</p> <p>Default value of this bit is 0b.</p> | <p>ECN: Base 6.3 RC-ECS$\triangleleft\triangleright$</p> |
| 2 | <p>ERR_FATAL/NONFATAL Received - Set when either a Fatal or a Non-fatal error Message is received and this bit is not already Set.</p> <p>Default value of this bit is 0b.</p> | RW1CS |
| 3 | <p>Multiple ERR_FATAL/NONFATAL Received - Set when either a Fatal or a Non-fatal error is received and <u>ERR_FATAL/NONFATAL Received</u> is already Set.</p> <p>Default value of this bit is 0b.</p> | RW1CS |
| 4 | <p>First Uncorrectable Fatal - Set when the first Uncorrectable error Message received is for a Fatal error.</p> <p>Default value of this field is 0b.</p> | RW1CS |
| 5 | <p>Non-Fatal Error Messages Received - Set when one or more Non-Fatal Uncorrectable error Messages have been received.</p> <p>Default value of this bit is 0b.</p> | RW1CS |
| 6 | <p>Fatal Error Messages Received - Set when one or more Fatal Uncorrectable error Messages have been received.</p> <p>Default value of this bit is 0b.</p> | RW1CS |
| 8:7 | <p>ERR_COR Subclass - If the Function is <u>ERR_COR Subclass</u> capable and the <u>ERR_COR Received</u> bit is not already Set, this field is loaded with the value of the <u>ERR_COR Subclass</u> field in If the If an applicable received <u>ERR_COR</u> Message. See § Section 2.2.8.3 . The value in this field is only valid when the <u>ERR_COR Received</u> bit is Set. If the Function is not <u>ERR_COR Subclass</u> capable, this field is Reserved.</p> <p>If the Function is <u>ERR_COR Subclass</u> capable and a <u>SIG_SFW</u> <u>ERR_COR</u> Message is received, system firmware should be signaled using a system-specific mechanism.</p> <p>If RC ECS Handling Capable is Set, other control bits determine which ERR COR Messages are applicable for logging by this field. See § Section 6.2.4.1.3.</p> <p>Default value of this field is 00b.</p> | <p>ROS / RsvdZ</p> |
| 9 | <p>If SFW Handling ERR COR Received – Set when an <u>ERR COR</u> Message being handled by <u>SFW</u> is received while this bit is not already Set. If RC ECS Handling Capable is Clear, this bit is RsvdZ.</p> | <p>If SFW Handling ERR COR Received – Set when an <u>ERR COR</u> Message being handled by <u>SFW</u> is received while this bit is not already Set. If RC ECS Handling Capable is Clear, this bit is RsvdZ.</p> |
| 10 | <p>If Multiple SFW Handling ERR COR Received – Set when an applicable <u>Correctable error Message</u> is received and <u>SFW Handling</u> <u>ERR COR Received</u> is already Set. If RC ECS Handling Capable is Clear, this bit is RsvdZ.</p> | <p>If Multiple SFW Handling ERR COR Received – Set when an applicable <u>Correctable error Message</u> is received and <u>SFW Handling</u> <u>ERR COR Received</u> is already Set. If RC ECS Handling Capable is Clear, this bit is RsvdZ.</p> |
| 31:27 | <p>Advanced Error Interrupt Message Number - When MSI/MSI-X is implemented, this register indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this Capability.</p> <p>For MSI, the value in this register indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this</p> | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the <u>Message Control Register for MSI</u>.</p> <p>For MSI-X, the value in this register indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this register must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this register must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this register is undefined.</p> | |

7.8.4.11 Error Source Identification Register (Offset 34h) §

The Error Source Identification Register identifies the source (Requester ID) of first 1↑applicable²⁰⁰ correctable and uncorrectable (Non-fatal/Fatal) errors reported in the Root Error Status Register. Refer to § Section 6.2 for further details. This register is updated regardless of the settings of the Root Control Register and the Root Error Command Register. § Figure 7-165 details allocation of register fields in the Error Source Identification Register ; § Table 7-146 provides the respective bit definitions.

For Functions other than Root Ports and Root Complex Event Collectors : when End-End TLP Prefix Supported is Set or Flit Mode Supported is Set, this register is RsvdP, otherwise this register is not required to be implemented.

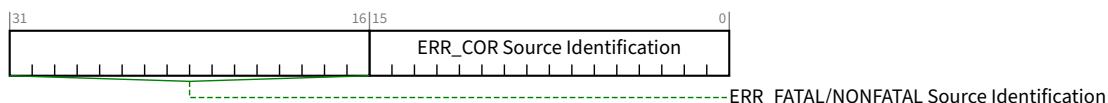


Figure 7-165 Error Source Identification Register §

Table 7-146 Error Source Identification Register §

| Bit Location | Register Description | Attributes |
|--------------|---|--|
| 15:0 | <p>ERR_COR Source Identification - Loaded with the Requester ID indicated <u>the</u> <u>1↑an applicable</u> received ERR_COR Message when the ERR_COR Received bit is not already set.</p> <p><u>1↑If RC ECS Handling Capable is Set, other control bits determine which ERR_COR Messages are applicable for logging by this field. See § Section 6.2.4.1.3.↑</u></p> <p>Default value of this field is 0000h.</p> | <p>ECN: Base 6.3 RC-ECS△↓</p> <p>ROS</p> |
| 31:16 | <p>ERR_FATAL/NONFATAL Source Identification - Loaded with the Requester ID indicated in the received ERR_FATAL or ERR_NONFATAL Message when the ERR_FATAL/NONFATAL Received bit is not already set.</p> <p>Default value of this field is 0000h.</p> | <p>ROS</p> |

200. 1↑To see which ERR_COR Messages are not applicable for this section, see § Section 6.2.4.1.3.↑

7.8.4.12 TLP Prefix Log Register (Offset 38h) §

The TLP Prefix Log Register captures the End-End TLP Prefix(s) for the TLP corresponding to the detected error; refer to § Section 6.2 for further details. The TLP Prefix Log Register is only meaningful when First Error Pointer is valid and the TLP Prefix Log Present bit is Set (see § Section 7.8.4.7).

The TLP Prefixes are captured such that, when read using DW accesses, the fields of the TLP Prefix are laid out in the same way the fields of the TLP Prefix are described. Therefore, byte 0 of a TLP Prefix is located in byte 3 of the associated TLP Prefix Log Register ; byte 1 of a TLP Prefix is located in byte 2; and so forth.

The First TLP Prefix Log Register contains the first End-End TLP Prefix from the TLP (see § Section 6.2.4.4). The Second TLP Prefix Log Register contains the second End-End TLP Prefix and so forth. If the TLP contains fewer than four End-End TLP Prefixes, the remaining TLP Prefix Log Registers contain zero. A TLP that contains more End-End TLP Prefixes than are indicated by the Function's Max End-End TLP Prefixes field must be handled as an error (see § Section 2.2.10.4 for specifics). To allow software to detect this condition, the supported number of End-End TLP Prefixes are logged in this register, the first overflow End-End TLP Prefix is logged in the first DW of the Header Log register and the remaining DWs of the Header Log register are undefined (see § Section 6.2.4.4).

The TLP Prefix Log Registers beyond the number supported by the Function are hardwired to zero. For example, if a Functions, Max End-End TLP Prefixes field contains 10b (indicating 2 DW of buffering) then the third and fourth TLP Prefix Log Registers are hardwired to zero. If the End-End TLP Prefix Supported bit (§ Section 7.5.3.15) is Clear, the TLP Prefix Log Register is not required to be implemented.

For VFs that share Header Log space, this register's contents are undefined when the Header Log contains all 1s due to an overflow condition. See § Section 6.2.4.2.1 for further requirements when VFs share Header Log space.

When Flit Mode Supported is Set and the link is operating in Flit Mode, this register is not present and the Header Log Register extends into this space (see additional DWs as indicated by the Header Log Size field).

When the link is operating in Non-Flit Mode, End-End TLP Prefixes are logged in the TLP Prefix Log Register .

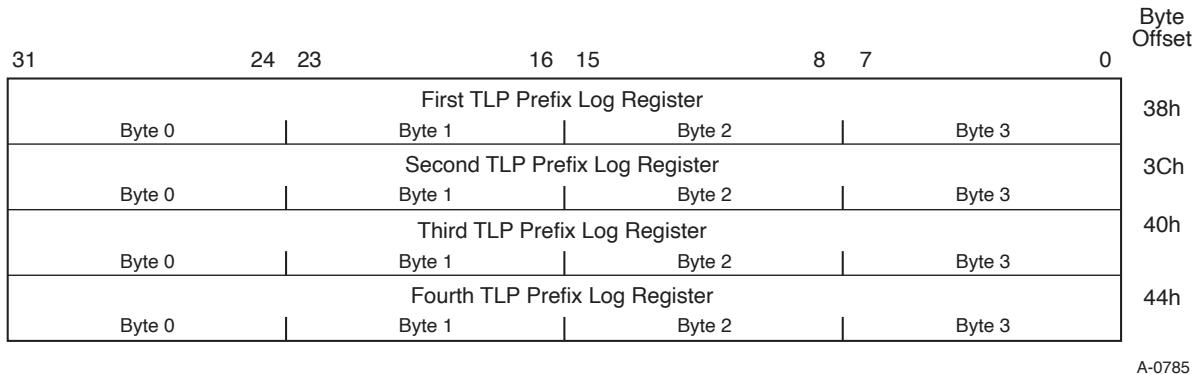


Figure 7-166 TLP Prefix Log Register §

Table 7-147 TLP Prefix Log Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|----------------------|------------|---------|
| 127:0 | TLP Prefix Log | ROS | 0 |

7.8.5 Enhanced Allocation Capability Structure (EA) §

Each function that supports the Enhanced Allocation mechanism must implement the Enhanced Allocation capability structure.

Each field is defined in the following sections. Reserved registers must return 0 when read and write operations must have no effect. Read-only registers return valid data when read, and write operations must have no effect.

7.8.5.1 Enhanced Allocation Capability First DW (Offset 00h) §

The first DW of the Enhanced Allocation capability is illustrated in § [Figure 7-167](#), and is documented in § [Table 7-148](#).

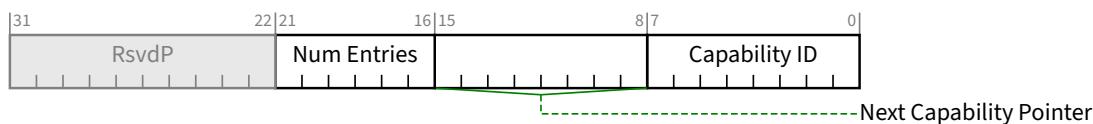


Figure 7-167 First DW of Enhanced Allocation Capability §

Table 7-148 First DW of Enhanced Allocation Capability §

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------|
| 7:0 | Capability ID - Must be set to 14h to indicate Enhanced Allocation capability. This field is read only. | HwInit |
| 15:8 | Next Capability Pointer - Pointer to the next item in the capabilities list. Must be NULL for the final item in the list. This field is read only. | HwInit |
| 21:16 | Num Entries - Number of entries following the first DW of the capability. Value of 00 0000b is permitted and means there are no entries. This field is read only. | HwInit |

7.8.5.2 Enhanced Allocation Capability Second DW (Offset 04h)

[Type 1 Functions Only] §

For Type 1 Functions only, there is a second DW in the capability, preceding the first entry. This second DW must be included in the Enhanced Allocation Capability whenever this capability is implemented in a Type 1 Function. The second DW of the Enhanced Allocation capability is illustrated in § [Figure 7-168](#), and is documented in § [Table 7-149](#).

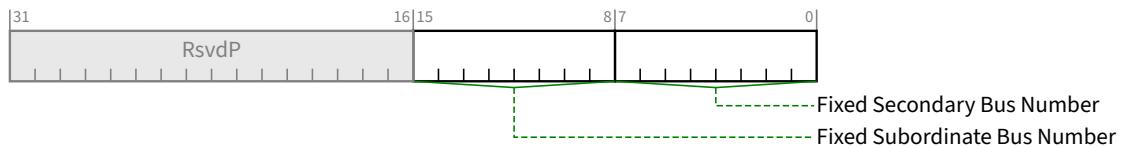


Figure 7-168 Second DW of Enhanced Allocation Capability §

Table 7-149 Second DW of Enhanced Allocation Capability §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Fixed Secondary Bus Number - If at least one Function that uses EA is located behind this Function, then this field must be set to indicate the Bus Number for the secondary interface of this Function. If no Function that uses EA is located behind this Function, then this field must be set to 00h. | HwInit |
| 15:8 | Fixed Subordinate Bus Number - If at least one Function that uses EA is located behind this Function, then this field must be set to indicate the highest Bus Number below this Function. If no Function that uses EA is located behind this Function, then this field must be set to 00h. | HwInit |

7.8.5.3 Enhanced Allocation Per-Entry Format (Offset 04h or 08h) §

An Enhanced Allocation Entry consists of a First DW followed by between 2 and 4 DW of Base / MaxOffset information.

- For Type 0 Functions , Enhanced Allocation Entries start at offset 04h of this capability.
- For Type 1 Functions , Enhanced Allocation Entries start at offset 08h of this capability.
- Subsequent Enhanced Allocation Entries immediately follow each other.

The first DW of each entry in the Enhanced Allocation capability is illustrated in § Figure 7-169 , and is defined in § Table 7-150 .

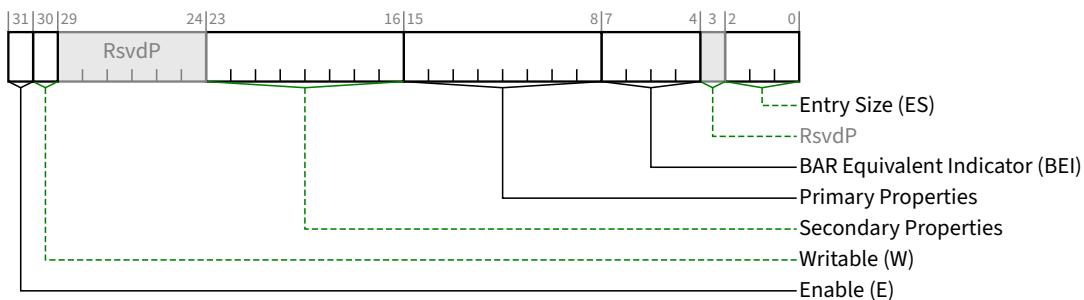


Figure 7-169 First DW of Each Entry for Enhanced Allocation Capability §

Table 7-150 First DW of Each Entry for Enhanced Allocation Capability §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|--|-------------|-------------|---|--|---|--|---|--|---|--|---|--|---|--|---|--|---|--------------------------|---|----------------------------|------|---|----|---|--------|
| 2:0 | <p>Entry Size (ES) - Number of DW following the initial DW in this entry.</p> <p>When processing this capability, software is required to use the value in this field to determine the size of this entry, and if this entry is not the final entry, the start of the following entry in the capability. This requirement must be strictly followed by software, even if the indicated entry size does not correspond to any entry defined in this specification.</p> <p>Value of 000b indicates only the first DW (containing the Entry Size field) is included in the entry.</p> | HwInit | | | | | | | | | | | | | | | | | | | | | | | | |
| 7:4 | <p>BAR Equivalent Indicator (BEI) - This field indicates the equivalent BAR for this entry.</p> <p>Specific rules for use of this field are given in the text following this table.</p> <table border="1"> <thead> <tr> <th>BEI Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0</td><td>Entry is equivalent to BAR at location 10h</td></tr> <tr><td>1</td><td>Entry is equivalent to BAR at location 14h</td></tr> <tr><td>2</td><td>Entry is equivalent to BAR at location 18h</td></tr> <tr><td>3</td><td>Entry is equivalent to BAR at location 1Ch</td></tr> <tr><td>4</td><td>Entry is equivalent to BAR at location 20h</td></tr> <tr><td>5</td><td>Entry is equivalent to BAR at location 24h</td></tr> <tr><td>6</td><td>Permitted to be used by a Function with a <u>Type 1 Configuration Space Header</u> only, optionally used to indicate a resource that is located <u>behind</u> the Function</td></tr> <tr><td>7</td><td>Equivalent Not Indicated</td></tr> <tr><td>8</td><td>Expansion ROM Base Address</td></tr> <tr><td>9-14</td><td>Entry relates to VF BARs 0-5 respectively</td></tr> <tr><td>15</td><td>Reserved - Software must treat values in this range as “Equivalent Not Indicated”</td></tr> </tbody> </table> | BEI Value | Description | 0 | Entry is equivalent to BAR at location 10h | 1 | Entry is equivalent to BAR at location 14h | 2 | Entry is equivalent to BAR at location 18h | 3 | Entry is equivalent to BAR at location 1Ch | 4 | Entry is equivalent to BAR at location 20h | 5 | Entry is equivalent to BAR at location 24h | 6 | Permitted to be used by a Function with a <u>Type 1 Configuration Space Header</u> only, optionally used to indicate a resource that is located <u>behind</u> the Function | 7 | Equivalent Not Indicated | 8 | Expansion ROM Base Address | 9-14 | Entry relates to VF BARs 0-5 respectively | 15 | Reserved - Software must treat values in this range as “Equivalent Not Indicated” | HwInit |
| BEI Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | Entry is equivalent to BAR at location 10h | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | Entry is equivalent to BAR at location 14h | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | Entry is equivalent to BAR at location 18h | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | Entry is equivalent to BAR at location 1Ch | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | Entry is equivalent to BAR at location 20h | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Entry is equivalent to BAR at location 24h | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | Permitted to be used by a Function with a <u>Type 1 Configuration Space Header</u> only, optionally used to indicate a resource that is located <u>behind</u> the Function | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | Equivalent Not Indicated | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | Expansion ROM Base Address | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9-14 | Entry relates to VF BARs 0-5 respectively | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | Reserved - Software must treat values in this range as “Equivalent Not Indicated” | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15:8 | Primary Properties - Indicates the entry properties as defined in § Table 7-152 . | HwInit | | | | | | | | | | | | | | | | | | | | | | | | |
| 23:16 | Secondary Properties - Optionally used to indicate a different but compatible entry property, using properties as defined in § Table 7-152 . | HwInit | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | <p>Writable (W) - The value 1b indicates that the <u>Base</u> and <u>MaxOffset</u> fields for this entry are <u>RW</u> and that the Field Size bits for this entry are either <u>RW</u> or <u>HwInit</u> . The value 0b indicates those fields are <u>HwInit</u> .</p> <p>See § Table 7-152 for additional requirements on the value of this field.</p> | HwInit | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | <p>Enable (E) - 1b indicates this entry is enabled, 0b indicates this entry is disabled.</p> <p>If system software disables this entry, the resource indicated must still be associated with this function, and it is not permitted to reallocate this resource to any other entity.</p> <p>This field is permitted to be implemented as <u>HwInit</u> for functions that require the allocation of the associated resource, or as <u>RW</u> for functions that can allow system software to disable this resource, for example if BAR mechanisms are to be used instead of this resource.</p> | RW / HwInit | | | | | | | | | | | | | | | | | | | | | | | | |

Rules for use of BEI field:

Base 6.4 vs Base 6.3

- A Type 0 Function is permitted to use EA to allocate resources for itself, and such resources must indicate a BEI value of 0-5, 7 or 8.
- **↑↑An SR-IOV↑** Physical Function (Type 0 Function that supports SR-IOV) is permitted to use EA to allocate resources for its associated Virtual Functions , and such resources must indicate a BEI value of 9-14. ECN: Base 6. SIOV△
- A Type 1 Function (bridge) is permitted to use EA to allocate resources for itself, and such resources must indicate a BEI value of 0, 1 or 7.
- A Type 1 Function is permitted but not required to indicate resources mapped behind that Function, but if such resources are indicated by the Type 1 Function , the entry must indicate a BEI value of 6.
- For a 64-bit Base Address Register , the BEI indicates the equivalent BAR location for lower DWORD.
- For Memory BARs where the Primary or Secondary Properties is 00h or 01h, it is permitted to assign the same BEI in the range of 0 to 5 once for a range where Base + MaxOffset is below 4 GB, and again for a range where Base + MaxOffset is greater than 4 GB; It is not otherwise permitted to assign the same BEI in the range 0 to 5 for more than one entry.
- For Virtual Function BARs where the Primary or Secondary Properties is 03h or 04h it is permitted to assign the same BEI in the range of 9 to 14 once for a range where Base + MaxOffset is below 4 GB, and again for a range where Base + MaxOffset is greater than 4 GB; It is not otherwise permitted to assign the same BEI in the range 9 to 14 for more than one VF entry.
- For all cases where two entries with the same BEI are permitted, Software must enable use of only one of the two ranges at a time for a given Function.
- It is permitted for an arbitrary number of entries to assign a BEI of 6 or 7.
- At most one entry is permitted with a BEI of 8; if such an entry is present, behavior of the Expansion ROM Base Address Register is changed (see § Section 7.5.1.2.4).
- For Type 1 Functions , BEI values 2 through 5 are reserved.

§ Figure 7-170 illustrates the format of a complete Enhanced Allocation entry for a Type 0 Function . For the Base and MaxOffset fields, bit 1 indicates if the field is a 32b (0) or 64b (1) field.

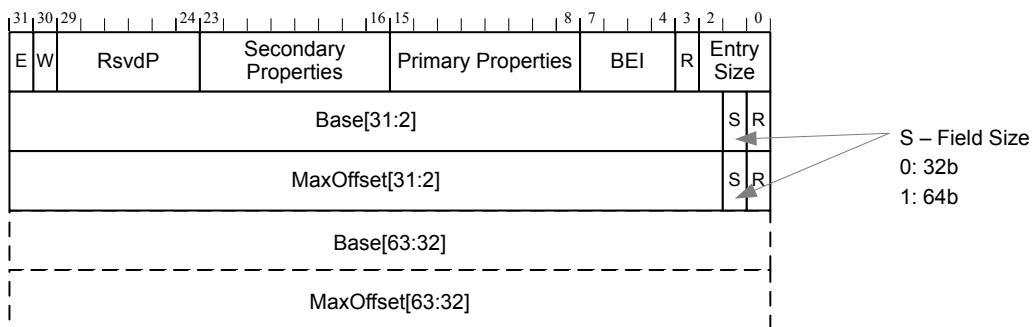


Figure 7-170 Format of Entry for Enhanced Allocation Capability §

The value in the **Base** field ([63:2] or [31:2]) indicates the DW address of the start of the resource range. Bits [1:0] of the address are not included in the **Base** field, and must always be interpreted as 00b.

The value in the **Base** field plus the value in the **MaxOffset** field ([63:2] or [31:2]) indicates the address of the last included DW of the resource range. Bits [1:0] of the **MaxOffset** are not included in the **MaxOffset** field, and must always be interpreted as 11b.

For the **Base** and **MaxOffset** fields, when bits [63:32] are not provided then those bits must be interpreted as all 0's.

Although it is permitted for a Type 0 Function to indicate the use of a range that is not naturally aligned and/or not a power of two in size, some system software may fail if this is done. Particularly for ranges that are mapped to legacy BARs by indicating a BEI in the range of 0 to 5, it is strongly recommended that the **Base** and **MaxOffset** fields for a Type 0 Function indicate a naturally aligned region.

The Primary Properties[7:0] field must be set by hardware to identify the type of resource indicated by the entry. It is strongly recommended that hardware set the Secondary Properties[7:0] to indicate an alternate resource type which can be used by software when the Primary Properties[7:0] field value is not comprehended by that software, for example when older system software is used with new hardware that implements resources using a value for Primary Properties that was reserved at the time the older system software was implemented. When this is done, hardware must ensure that software operating using the resource according to the value indicated in the Secondary Properties field will operate in a functionally correct way, although it is not required that this operation will result in optimal system performance or behavior.

The Primary Properties[7:0] and Secondary Properties[7:0] fields are defined in § Table 7-152 . This table also defines whether or not the entry is permitted to be writeable. The Writeable bit in any entry must be 0b unless both the Primary and Secondary properties of that entry allow otherwise.

Table 7-152 Enhanced Allocation Entry Field Value Definitions for both the Primary Properties and Secondary Properties Fields §

| Value (h) | Resource Definition | Writeable permitted |
|-----------|--|---------------------|
| 00-01 | Memory Space | No |
| 02 | I/O Space. | No |
| 03-04 | For use only by Physical Functions to indicate resources for Virtual Function use, Memory Space. | No |
| 05-06 | For use only by Type 1 Functions to indicate Memory, for Allocation Behind that Bridge. | No |
| 07 | For use only by Type 1 Functions to indicate I/O Space for Allocation Behind that Bridge. | No |
| 08-FC | Reserved for future use; System firmware/software must not write to this entry, and must not attempt to interpret this entry or to use this resource. When software reads a Primary Properties value that is within this range, is it strongly recommended that software treat this resource according to the value in the Secondary Properties field, if that field contains a non-reserved value. | Yes |
| FD | Memory Space Resource Unavailable For Use - System firmware/software must not write to this entry, and must not attempt to use the resource described by this entry for any purpose. | No |
| FE | I/O Space Resource Unavailable For Use - System firmware/software must not write to this entry, and must not attempt to use the resource described by this entry for any purpose. | No |
| FF | Entry Unavailable For Use - System firmware/software must not write to this entry, and must not attempt to interpret this entry as indicating any resource. | No |

| Value (h) | Resource Definition | Writable permitted |
|-----------|--|-----------------------|
| | Hardware <i>MUST</i> @FLIT use this value in the Secondary Properties field to indicate that, for proper operation, the hardware requires the use of the resource definition indicated in the Primary Properties field | . |

The following figures illustrate the layout of Enhanced Allocation entries for various cases.

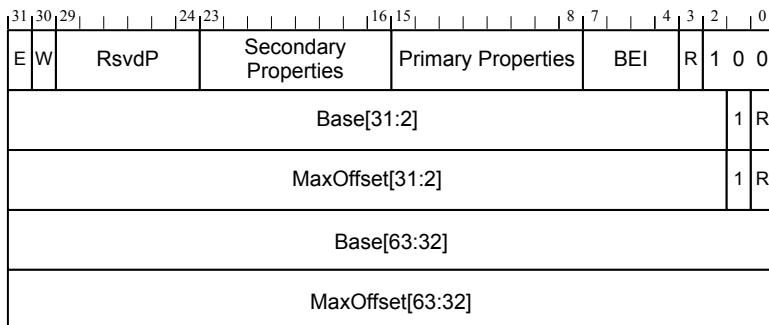


Figure 7-171 Example Entry with 64b Base and 64b MaxOffset §

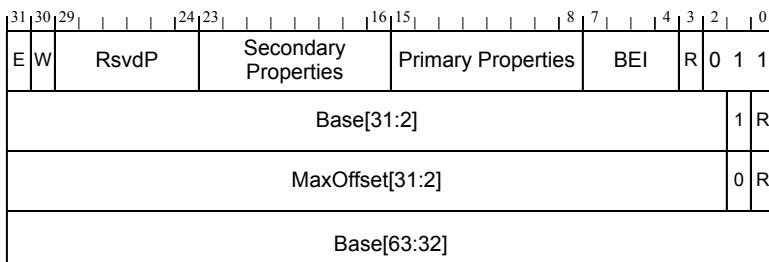


Figure 7-172 Example Entry with 64b Base and 32b MaxOffset §

Base 6.4 vs Base 6.3

| | | | | | | | | | | | | | | | | | |
|------------------|----|-------|----------------------|--------------------|-----|---|----|----|---|---|---|--|---|---|---|---|-----|
| 31 | 30 | 29 | | 24 | 23 | | 16 | 15 | | 8 | 7 | | 4 | 3 | 2 | 1 | 0 |
| E | W | RsvdP | Secondary Properties | Primary Properties | BEI | R | 0 | 1 | 1 | | | | | | | | |
| Base[31:2] | | | | | | | | | | | | | | | | | 0 R |
| MaxOffset[31:2] | | | | | | | | | | | | | | | | | 1 R |
| MaxOffset[63:32] | | | | | | | | | | | | | | | | | |

Figure 7-173 Example Entry with 32b Base and 64b MaxOffset §

| | | | | | | | | | | | | | | | | | |
|-----------------|----|-------|----------------------|--------------------|-----|---|----|----|---|---|---|--|---|---|---|---|-----|
| 31 | 30 | 29 | | 24 | 23 | | 16 | 15 | | 8 | 7 | | 4 | 3 | 2 | 1 | 0 |
| E | W | RsvdP | Secondary Properties | Primary Properties | BEI | R | 0 | 1 | 0 | | | | | | | | |
| Base[31:2] | | | | | | | | | | | | | | | | | 0 R |
| MaxOffset[31:2] | | | | | | | | | | | | | | | | | 0 R |

Figure 7-174 Example Entry with 32b Base and 32b MaxOffset §

7.8.6 Resizable BAR Extended Capability §

The Resizable BAR Extended Capability is an optional capability that allows hardware to communicate resource sizes, and system software, after determining the optimal size, to communicate this optimal size back to the hardware. Hardware communicates the resource sizes that are acceptable for operation via the Resizable BAR Capability and Control registers. Hardware must support at least one size in the range from 1 MB to 512 GB.

IMPLEMENTATION NOTE: RESIZABLE BAR BACKWARD COMPATIBILITY WITH SOFTWARE §

The Resizable BAR Extended Capability initially supported 20 sizes, ranging from 1 MB to 512 GB, and was later expanded with 16 larger sizes. The hardware requirement to support at least one of the initial sizes ensures backward compatibility with software that comprehends only the initial sizes.

Software determines, through a proprietary mechanism, what the optimal size is for the resource, and programs that size via the BAR Size field of the Resizable BAR Control register. Hardware immediately reflects the size inference in the

read-only bits of the appropriate Base Address register. Hardware must Clear any bits that change from RW to read-only, so that subsequent reads return zero. Software must clear the Memory Space Enable bit in the Command register before writing the BAR Size field. After writing the BAR Size field, the contents of the corresponding BAR are undefined. To ensure that it contains a valid address after resizing the BAR, system software must reprogram the BAR, and Set the Memory Space Enable bit (unless the resource is not allocated).

The Resizable BAR Capability and Control registers are permitted to indicate the ability to operate at 4 GB or greater only if the associated BAR is a 64-bit BAR.

This capability is applicable to Functions that have Base Address registers only. The capability is permitted to be present in PFs. Since VFs do not implement standard BARs, the capability must not be present in a VF. The PF's Resizable BAR settings do not affect any settings in the SR-IOV Extended Capability.

It is strongly recommended that a Function not advertise any supported BAR sizes that are larger than the space it would effectively utilize if allocated.

IMPLEMENTATION NOTE: USING THE CAPABILITY DURING RESOURCE ALLOCATION §

System software that allocates resources can use this capability to resize the resources inferred by the Function's BAR's read-only bits. Previous versions of this software determined the resource size by writing FFFF_FFFFh or FFFF_FFFF_FFFFh to the BAR, reading back the value, and determining the size by the number of bits that are Set. Following this, the base address is written to the BAR.

System software uses this capability in place of the above mentioned method of determining the resource size, and prior to assigning the base address to the BAR. Potential usable resource sizes are reported by the Function via its Resizable BAR Capability and Control registers. It is intended that the software allocate the largest of the reported sizes that it can, since allocating less address space than the largest reported size can result in lower performance. Software then writes the size to the Resizable BAR Control register for the appropriate BAR for the Function. Following this, the base address is written to the BAR.

For interoperability reasons, it is possible that hardware will set the default size of the BAR to a low size; that is, a size lower than the largest reported in the Resizable BAR Capability and Control registers. Software that does not use this capability to size resources will likely result in sub-optimal resource allocation, where the resources are smaller than desirable, or not allocatable because there is no room for them.

With the Resizable BAR capability, the amount of address space consumed by a device can change. In a resource constrained environment, the allocation of more address space to a device may result in allocation of less of the address space to other memory-mapped hardware, like system RAM. System software responsible for allocating resources in this kind of environment is recommended to distribute the limited address space appropriately.

The Resizable BAR Capability structure defines a PCI Express Extended Capability, which is located in PCI Express Extended Configuration Space, that is, above the first 256 bytes, and is shown below in § Figure 7-175. This structure allows devices with this capability to be identified and controlled. A Capability and a Control register is implemented for each BAR that is resizable. Since a maximum of six BARs may be implemented by any Function, the Resizable BAR Capability structure can range from 12 bytes long (for a single BAR) to 52 bytes long (for all six BARs).

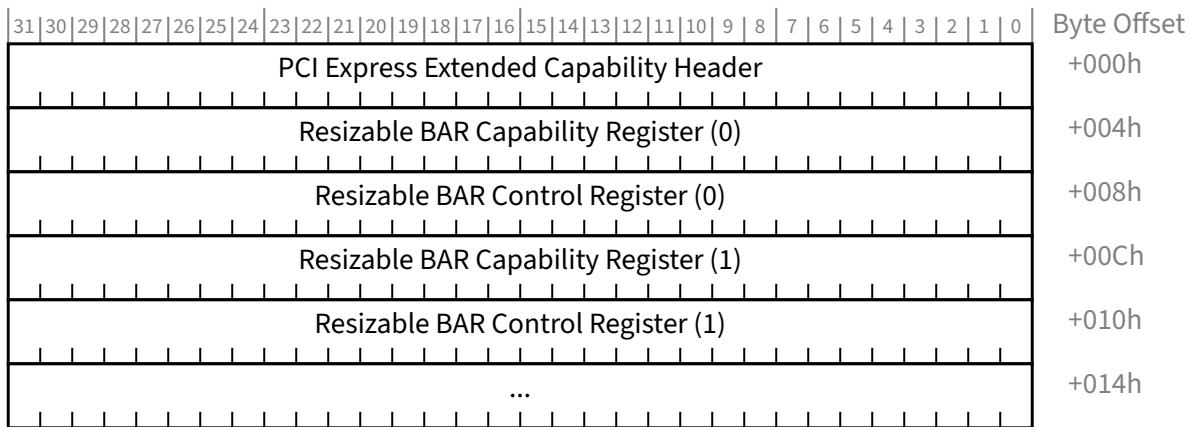


Figure 7-175 Resizable BAR Extended Capability §

7.8.6.1 Resizable BAR Extended Capability Header (Offset 00h) §

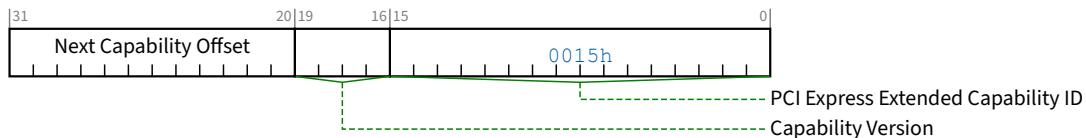


Figure 7-176 Resizable BAR Extended Capability Header §

Table 7-153 Resizable BAR Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. The PCI Express Extended Capability ID for the Resizable BAR Capability is 0015h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

7.8.6.2 Resizable BAR Capability Register §

For backward compatibility with software, hardware must Set at least one bit in the range from 4 to 23. See the associated Implementation Note in § Section 7.8.6 .

Base 6.4 vs Base 6.3

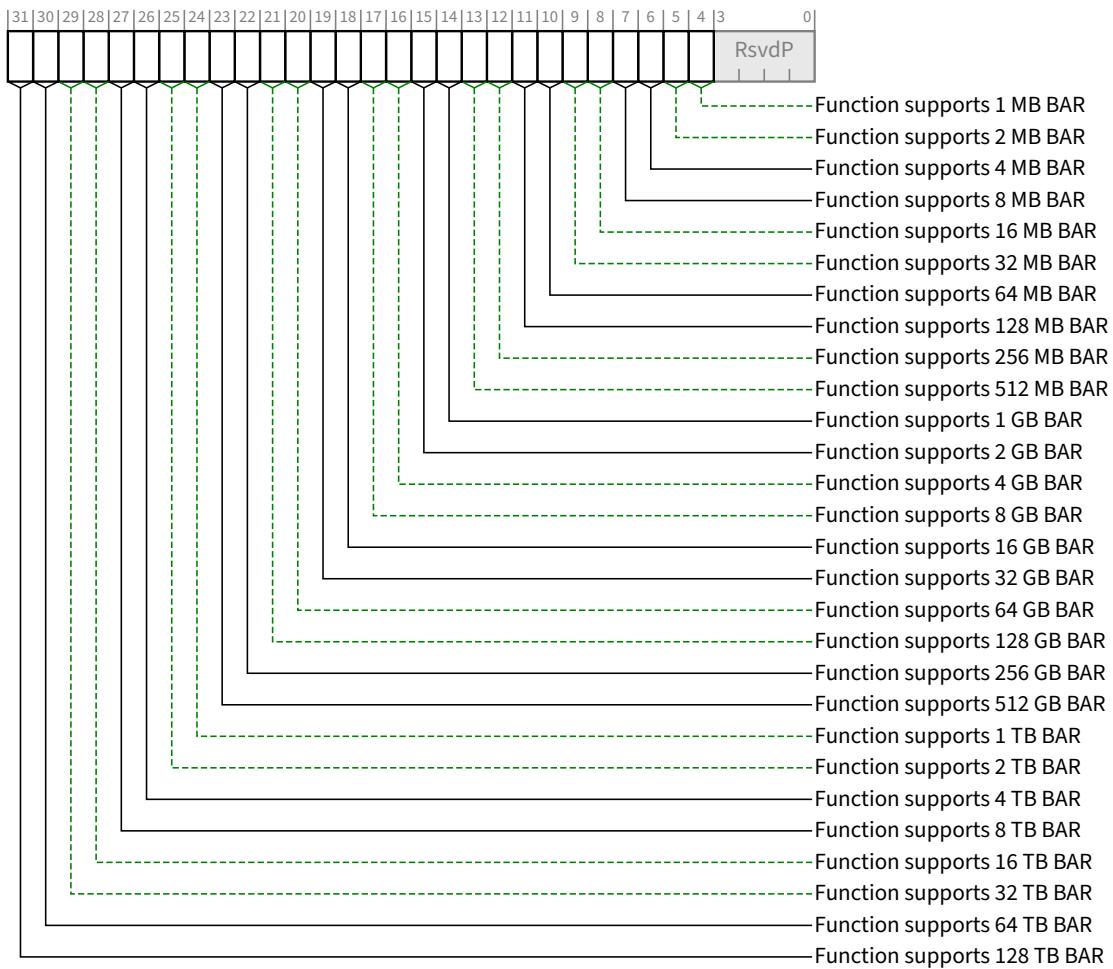


Figure 7-177 Resizable BAR Capability Register §

Table 7-154 Resizable BAR Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 4 | Function supports 1 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 1 MB (2^{20} bytes) | RO |
| 5 | Function supports 2 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 2 MB (2^{21} bytes) | RO |
| 6 | Function supports 4 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 4 MB (2^{22} bytes) | RO |
| 7 | Function supports 8 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 8 MB (2^{23} bytes) | RO |
| 8 | Function supports 16 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 16 MB (2^{24} bytes) | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 9 | Function supports 32 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 32 MB (2^{25} bytes) | RO |
| 10 | Function supports 64 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 64 MB (2^{26} bytes) | RO |
| 11 | Function supports 128 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 128 MB (2^{27} bytes) | RO |
| 12 | Function supports 256 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 256 MB (2^{28} bytes) | RO |
| 13 | Function supports 512 MB BAR - When Set, indicates that the Function supports operating with the BAR sized to 512 MB (2^{29} bytes) | RO |
| 14 | Function supports 1 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 1 GB (2^{30} bytes) | RO |
| 15 | Function supports 2 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 2 GB (2^{31} bytes) | RO |
| 16 | Function supports 4 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 4 GB (2^{32} bytes) | RO |
| 17 | Function supports 8 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 8 GB (2^{33} bytes) | RO |
| 18 | Function supports 16 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 16 GB (2^{34} bytes) | RO |
| 19 | Function supports 32 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 32 GB (2^{35} bytes) | RO |
| 20 | Function supports 64 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 64 GB (2^{36} bytes) | RO |
| 21 | Function supports 128 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 128 GB (2^{37} bytes) | RO |
| 22 | Function supports 256 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 256 GB (2^{38} bytes) | RO |
| 23 | Function supports 512 GB BAR - When Set, indicates that the Function supports operating with the BAR sized to 512 GB (2^{39} bytes) | RO |
| 24 | Function supports 1 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 1 TB (2^{40} bytes) | RO |
| 25 | Function supports 2 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 2 TB (2^{41} bytes) | RO |
| 26 | Function supports 4 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 4 TB (2^{42} bytes) | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 27 | Function supports 8 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 8 TB (2^{43} bytes) | RO |
| 28 | Function supports 16 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 16 TB (2^{44} bytes) | RO |
| 29 | Function supports 32 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 32 TB (2^{45} bytes) | RO |
| 30 | Function supports 64 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 64 TB (2^{46} bytes) | RO |
| 31 | Function supports 128 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 128 TB (2^{47} bytes) | RO |

Base 6.4 vs Base 6.3

7.8.6.3 Resizable BAR Control Register §

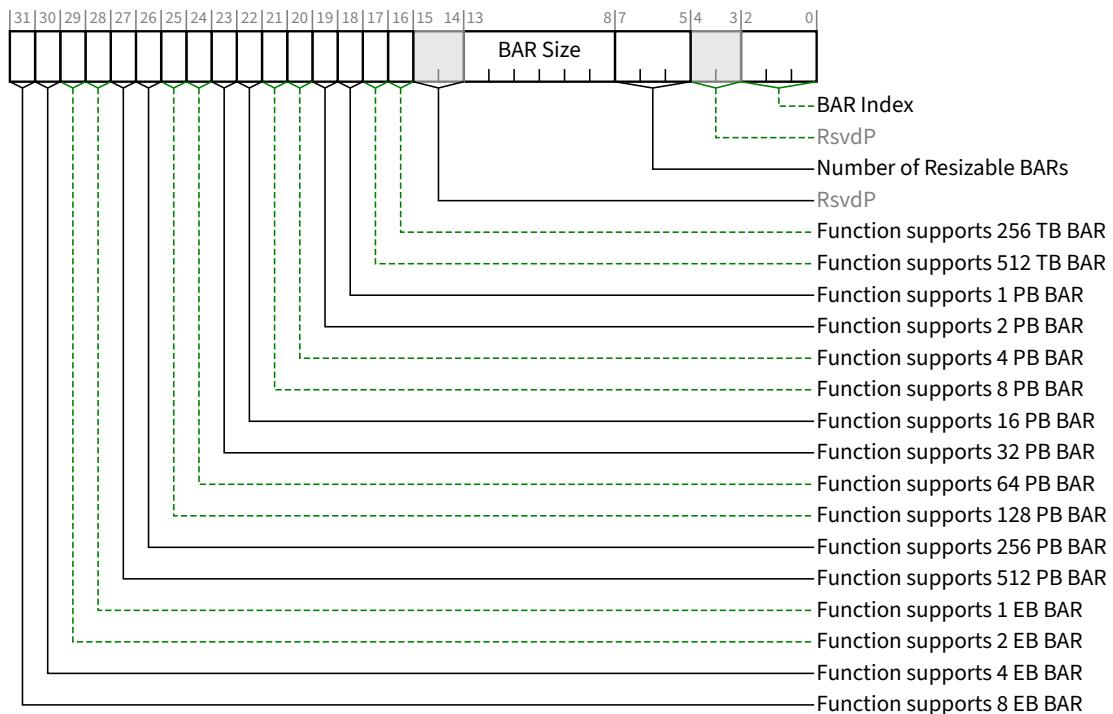


Figure 7-178 Resizable BAR Control Register §

Table 7-155 Resizable BAR Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | BAR Index - This encoded value points to the beginning of the BAR. 0 BAR located at offset 10h | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>1 BAR located at offset 14h</p> <p>2 BAR located at offset 18h</p> <p>3 BAR located at offset 1Ch</p> <p>4 BAR located at offset 20h</p> <p>5 BAR located at offset 24h</p> <p>Others All other encodings are Reserved.</p> <p>For a 64-bit Base Address register, the BAR Index indicates the lower DWORD.</p> <p>This value indicates which BAR supports a negotiable size.</p> | |
| 7:5 | <p>Number of Resizable BARs - Indicates the total number of resizable BARs in the capability structure for the Function. See § Figure 7-175.</p> <p>The value of this field must be in the range of 01h to 06h. The field is valid in Resizable BAR Control register (0) (at offset 008h), and is RsvdP for all others.</p> | RO / RsvdP |
| 13:8 | <p>BAR Size - This is an encoded value.</p> <p>0 1 MB (2^{20} bytes)</p> <p>1 2 MB (2^{21} bytes)</p> <p>2 4 MB (2^{22} bytes)</p> <p>3 8 MB (2^{23} bytes)</p> <p>...</p> <p>43 8 EB (2^{63} bytes)</p> <p>The default value of this field is equal to the default size of the address space that the BAR resource is requesting via the BAR's read-only bits. For backward compatibility with software, the default value must be in the range from 0 to 19.</p> <p>When this register field is programmed, the value is immediately reflected in the size of the resource, as encoded in the number of read-only bits in the BAR.</p> <p>Software must only write values that correspond to those indicated as supported in the Resizable BAR Capability and Control registers. Writing an unsupported value will produce undefined results. BAR Size bits that never need to be Set in order to indicate every supported size are permitted to be hardwired to 0.</p> | RW |
| 16 | Function supports 256 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 256 TB (2^{48} bytes) | RO |
| 17 | Function supports 512 TB BAR - When Set, indicates that the Function supports operating with the BAR sized to 512 TB (2^{49} bytes) | RO |
| 18 | Function supports 1 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 1 PB (2^{50} bytes) | RO |
| 19 | Function supports 2 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 2 PB (2^{51} bytes) | RO |
| 20 | Function supports 4 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 4 PB (2^{52} bytes) | RO |
| 21 | Function supports 8 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 8 PB (2^{53} bytes) | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 22 | Function supports 16 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 16 PB (2^{54} bytes) | RO |
| 23 | Function supports 32 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 32 PB (2^{55} bytes) | RO |
| 24 | Function supports 64 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 64 PB (2^{56} bytes) | RO |
| 25 | Function supports 128 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 128 PB (2^{57} bytes) | RO |
| 26 | Function supports 256 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 256 PB (2^{58} bytes) | RO |
| 27 | Function supports 512 PB BAR - When Set, indicates that the Function supports operating with the BAR sized to 512 PB (2^{59} bytes) | RO |
| 28 | Function supports 1 EB BAR - When Set, indicates that the Function supports operating with the BAR sized to 1 EB (2^{60} bytes) | RO |
| 29 | Function supports 2 EB BAR - When Set, indicates that the Function supports operating with the BAR sized to 2 EB (2^{61} bytes) | RO |
| 30 | Function supports 4 EB BAR - When Set, indicates that the Function supports operating with the BAR sized to 4 EB (2^{62} bytes) | RO |
| 31 | Function supports 8 EB BAR - When Set, indicates that the Function supports operating with the BAR sized to 8 EB (2^{63} bytes) | RO |

7.8.7 VF Resizable BAR Extended Capability §

The VF Resizable BAR Extended Capability is permitted to be implemented only in PFs that implement at least one VF BAR, and affects the size and base of a PF's VF BARs. Since VFs do not implement the BARs themselves the capability must not be present in a VF. A PF may implement both a VF Resizable BAR Extended Capability and a Resizable BAR capability, as each capability operates independently.

The VF Resizable BAR Extended Capability is an optional capability that permits PFs to be able to have their VF's BARs resized. The VF Resizable BAR Extended Capability permits hardware to communicate the resource sizes that are acceptable for operation via the VF Resizable BAR Extended Capability and Control registers and system software to communicate the optimal size back to the hardware via the VF BAR Size field of the VF Resizable BAR Control register.

Hardware immediately reflects the size inference in the read-only bits of the appropriate VF BAR. The size inferred is the greater of the values decoded from the System Page Size and VF BAR Size fields. Hardware must Clear any bits that change from read-write to read-only, so that subsequent reads return zero. Software must clear the VF MSE bit in the SR-IOV Control Register before writing the VF BAR Size field. After writing the VF BAR Size field, the contents of the corresponding VF BAR are undefined. To ensure that it contains a valid address after resizing the VF BAR, system software must reprogram the VF BAR, and Set the VF MSE bit (unless the resource is not allocated).

The VF Resizable BAR Extended Capability and Control registers are permitted to indicate the ability to operate at 4 GB or greater only if the associated VF BAR is a 64-bit BAR.

It is strongly recommended that a Function not advertise any supported VF BAR size values in this capability that are larger than the space it would effectively utilize if allocated.

IMPLEMENTATION NOTE: USING THE CAPABILITY DURING RESOURCE ALLOCATION §

System software uses this capability in a similar way to the Resizable BAR capability. System software must first configure the System Page Size register (see § Section 9.2.1.1.1). Potential usable memory aperture sizes are reported by the PF, and read, from the VF Resizable BAR Extended Capability and Control registers. It is intended that the software allocate the largest of the reported sizes that it can, since allocating less address space than the largest reported size can result in lower performance. Software then writes the size to the VF Resizable BAR Control register for the appropriate VF BAR for the Function. Following this, the base address is written to the VF BAR.

For interoperability reasons, it is possible that hardware will set the default size of the VF BAR to a low size; a size lower than the largest reported in the VF Resizable BAR Capability Register . Software that does not use this capability to size resources will likely result in sub-optimal resource allocation, where the resources are smaller than desirable, or not allocatable because there is no room for them. It is recommended that system software responsible for allocating resources in a resource constrained environment distribute the limited address space to all memory-mapped hardware, including system RAM, appropriately.

The VF Resizable BAR Extended Capability structure defines a PCI Express Extended Capability which is located in PCI Express Extended Configuration Space, that is, above the first 256 bytes, and is shown below in § Figure 7-179 . This structure allows PFs with this capability to be identified and controlled. A Capability register and a Control register are implemented for each VF BAR that is resizable. Since a maximum of 6 VF BARs may be implemented by any PF, the VF Resizable BAR Capability structure can range from 12 bytes long (for a single VF BAR) to 52 bytes long (for all 6 VF BARs).

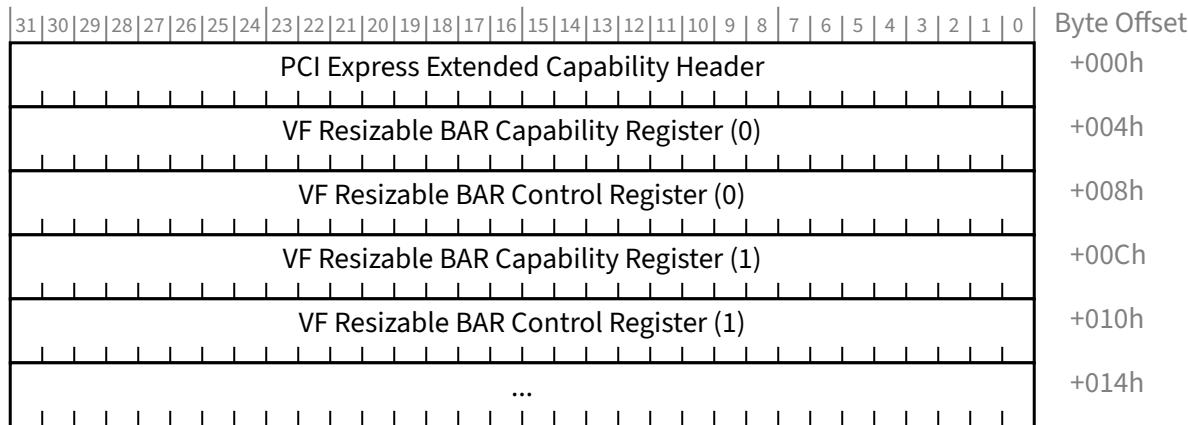


Figure 7-179 VF Resizable BAR Extended Capability §

7.8.7.1 VF Resizable BAR Extended Capability Header (Offset 00h) §

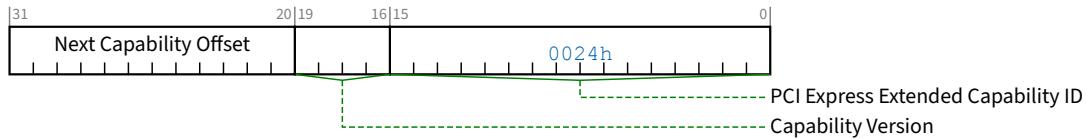


Figure 7-180 VF Resizable BAR Extended Capability Header §

Table 7-156 VF Resizable BAR Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | <p>PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability.</p> <p>PCI Express Extended Capability ID for the VF Resizable BAR Extended Capability is 0024h .</p> | RO |
| 19:16 | <p>Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present.</p> <p>Must be 1h for this version of the specification.</p> | RO |
| 31:20 | <p>Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities</p> | RO |

7.8.7.2 VF Resizable BAR Capability Register (Offset 04h) §

The VF Resizable BAR Capability Register field descriptions are the same as the definitions in the Resizable BAR Capability Register in § Table 7-154 . Where those descriptions say ‘BAR’, this register’s description is for ‘VF BAR’. Where those descriptions say ‘Function’, this register’s description is for ‘PF’. Otherwise, the field descriptions, the number of bits, their positions, and their attributes are the same. Consequently § Figure 7-177 similarly allocates the register fields in this register.

7.8.7.3 VF Resizable BAR Control Register (Offset 08h) §

The VF Resizable BAR Control register bits 31:16 follow the same definitions as the Resizable BAR Control register in § Table 7-155 .

Base 6.4 vs Base 6.3

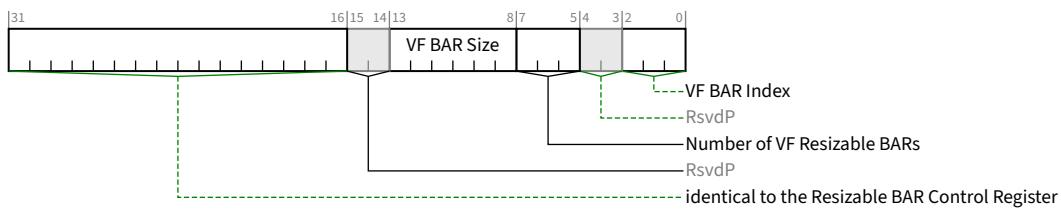


Figure 7-181 VF Resizable BAR Control Register §

Table 7-157 VF Resizable BAR Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>VF BAR Index - This encoded value points to the beginning of this particular VF BAR located in the SR-IOV Extended Capability.</p> <ul style="list-style-type: none"> 0 VF BAR located at offset 24h 1 VF BAR located at offset 28h 2 VF BAR located at offset 2Ch 3 VF BAR located at offset 30h 4 VF BAR located at offset 34h 5 VF BAR located at offset 38h others All other encodings are reserved. <p>For a 64-bit Base Address register, the <u>VF BAR Index</u> indicates the lower DWORD.</p> <p>This value indicates which VF BAR supports a negotiable size.</p> | RO |
| 7:5 | <p>Number of VF Resizable BARs - Indicates the total number of resizable VF BARs in the capability structure for the Function. See § Figure 7-179.</p> <p>The value of this field must be in the range of 01h to 06h. The field is valid in VF Resizable BAR Control register (0) (at offset 08h), and is RsvdP for all others.</p> | RO / RsvdP |
| 13:8 | <p>VF BAR Size - This is an encoded value.</p> <ul style="list-style-type: none"> 0 1 MB (2^{20} bytes) 1 2 MB (2^{21} bytes) 2 4 MB (2^{22} bytes) 3 8 MB (2^{23} bytes) ... 43 8 EB (2^{63} bytes) <p>The default value of this field is equal to the default size of the address space that the VF BAR resource is requesting via the VF BAR's read-only bits.</p> <p>Software must only write values that correspond to those indicated as supported in the VF Resizable BAR Capability and Control registers. Writing an unsupported value will produce undefined results.</p> <p>When this register field is programmed, the value is immediately reflected in the size of the resource, as encoded in the number of read-only bits in the VF BAR.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|---|--------------------------|
| 31:16 | These bits are identical to the Resizable BAR Control Register bits [31:16] defined in § Figure 7-178 . Where those descriptions say ‘BAR’, this register’s description is for ‘VF BAR’. Where those descriptions say ‘Function’, this register’s description is for ‘PF’. | See § Figure 7-178 |

7.8.8 ARI Extended Capability §

ARI is an optional capability, except as stated below. This capability must be implemented by each Function in an ARI Device . It is not applicable to a Root Port , a Switch Downstream Port, an RCiEP , or a Root Complex Event Collector .

For **↑↓SR-IOV devices** **↑↓SR-IOV Devices** not in a Root Complex, implementing the ARI Extended Capability in each Function is mandatory.

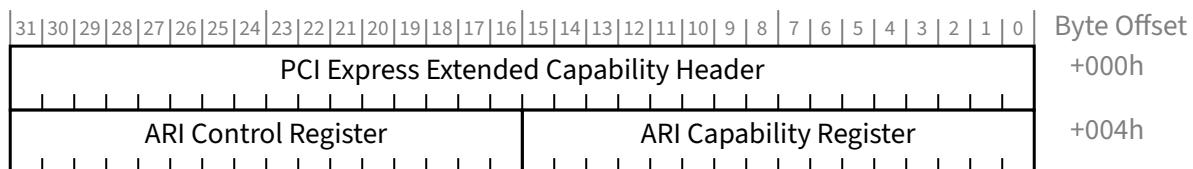


Figure 7-182 ARI Extended Capability §

7.8.8.1 ARI Extended Capability Header (Offset 00h) §

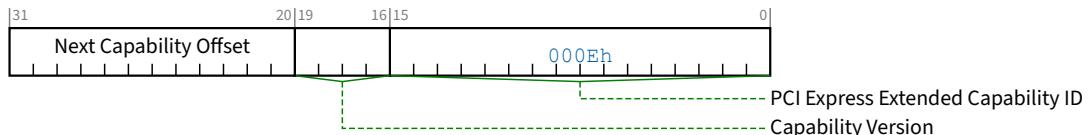


Figure 7-183 ARI Extended Capability Header §

Table 7-158 ARI Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the ARI Extended Capability is 000Eh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

7.8.8.2 ARI Capability Register (Offset 04h) §

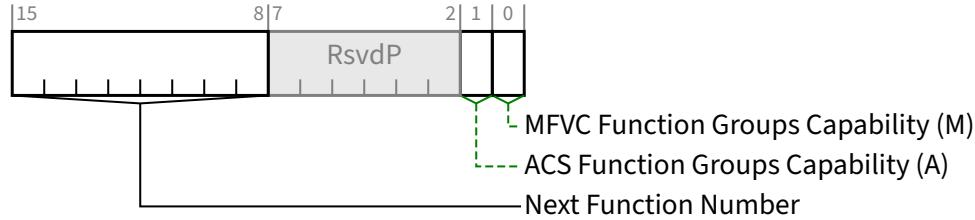


Figure 7-184 ARI Capability Register §

Table 7-159 ARI Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>MFVC Function Groups Capability (M) - Applicable only for Function 0; must be Zero for all other Functions. If Set, indicates that the ARI Device supports Function Group level arbitration via its Multi-Function Virtual Channel (MFVC) Capability structure.</p> <p>Any ↓↑SR-IOV Device↓↑↑SR-IOV Device↑ that implements the MFVC Extended Capability with the optional Function Arbitration Table and consumes more than one Bus Number must Set this bit in its Function 0.</p> | RO |
| 1 | <p>ACS Function Groups Capability (A) - Applicable only for Function 0; must be Zero for all other Functions. If Set, indicates that the ARI Device supports Function Group level granularity for ACS P2P Egress Control via its ACS Capability structures.</p> <p>Any ↓↑SR-IOV Device↓↑↑SR-IOV Device↑ that implements the ACS Capability with the optional Egress Control Vector and consumes more than one Bus Number must Set this bit in its Function 0.</p> | RO |
| 15:8 | <p>Next Function Number - With non-VFs, this field indicates the Function Number of the next higher numbered Function in the Device, or 00h if there are no higher numbered Functions. Function 0 starts this linked list of Functions.</p> <p>The presence of ↓↑Shadow Functions↓↑↑Shadow Functions↑ does not affect this field.</p> <p>For VFs, this field is undefined since VFs are located using First VF Offset (see § Section 9.4.3.9) and VF Stride (see § Section 9.4.3.10).</p> | RO |

7.8.8.3 ARI Control Register (Offset 06h) §

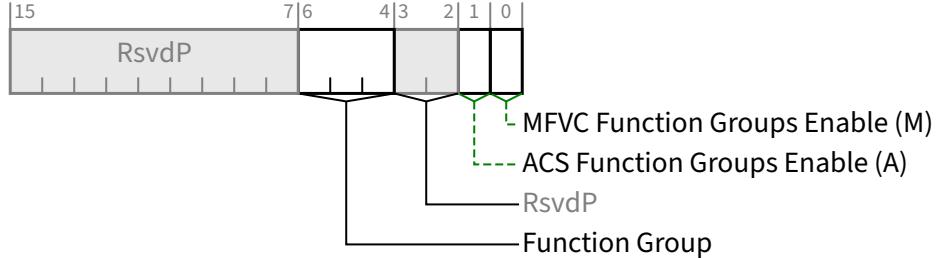


Figure 7-185 ARI Control Register §

Table 7-160 ARI Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>MFVC Function Groups Enable (M) - Applicable only for Function 0; must be hardwired to 0b for all other Functions. When set, the ARI Device must interpret entries in its Function Arbitration Table as Function Group Numbers rather than Function Numbers.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the MFVC Function Groups Capability bit is 0b.</p> | RW |
| 1 | <p>ACS Function Groups Enable (A) - Applicable only for Function 0; must be hardwired to 0b for all other Functions. When set, each Function in the ARI Device must associate bits within its Egress Control Vector with Function Group Numbers rather than Function Numbers.</p> <p>Default value of this bit is 0b. Must be hardwired to 0b if the ACS Function Groups Capability bit is 0b.</p> | RW |
| 6:4 | <p>Function Group - Assigns a Function Group Number to this Function.</p> <p>Default value of this field is 000b. Must be hardwired to 000b if in Function 0, the MFVC Function Groups Capability bit and ACS Function Groups Capability bit are both 0b.</p> | RW |

7.8.9 PASID Extended Capability Structure §

The presence of a PASID Extended Capability indicates that one or more Endpoint Functions support sending and receiving TLPs containing a PASID TLP Prefix in NFM or containing OHC with PASID in FM. Separate support and enables are provided for the various optional features.

This capability is only permitted to be implemented in to Endpoints Functions (including RCiEPs). A PF is permitted to implement the PASID capability, but VFs must not implement it. For Root Ports, support and control is outside the scope of this specification.

In earlier revisions of this specification, it was ambiguous whether a single PASID capability or multiple PASID capabilities are supported for an MFD. For backward compatibility, both models are supported. In the PASID Capability and Control register bit definitions, the phrase “applicable Endpoint Function” refers to the following rules:

- If an SFD contains a PASID capability, it must apply to the SFD’s single Endpoint Function.
- If an MFD contains a single PASID capability, it must apply to all Endpoint Functions in the MFD, including all VFs regardless of their PF association.

- If an MFD contains multiple PASID capabilities, each PASID capability that exists must apply only to the Endpoint Function in which it resides, with the exception that if a PF contains a PASID capability, that PASID capability must apply to all VFs associated with that PF.

When a PASID capability applies to multiple Endpoint Functions in a device, the device sends the requesting Function's ID in the Requester ID field of the TLP containing the PASID.

This capability is independent of both the ATS and PRI features defined in § Chapter 10.. Endpoint Functions that contain a PASID Extended Capability need not support ATS or PRI. Endpoint Functions that support ATS or PRI need not support PASID.

§ Figure 7-186 details allocation of the register bits in the PASID Extended Capability structure.

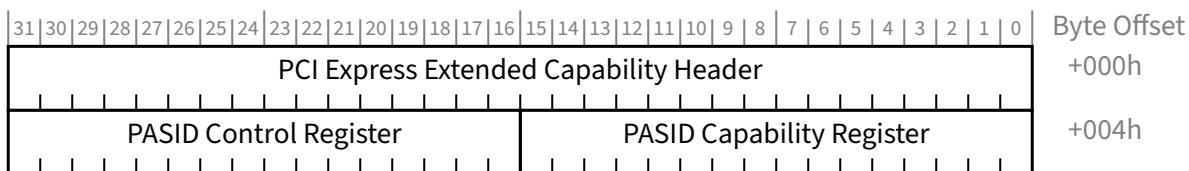


Figure 7-186 PASID Extended Capability Structure §

7.8.9.1 PASID Extended Capability Header (Offset 00h) §

§ Figure 7-187 details allocation of the register fields in the PASID Extended Capability Header ; § Table 7-161 provides the respective bit definitions.

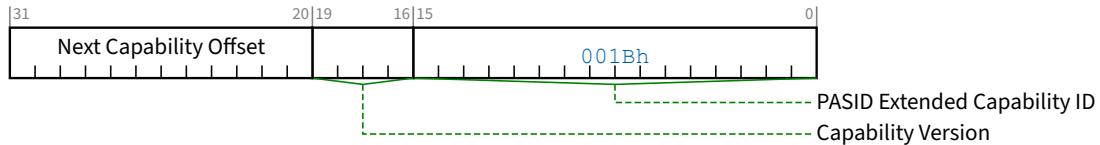


Figure 7-187 PASID Extended Capability Header §

Table 7-161 PASID Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PASID Extended Capability ID - Indicates the PASID Extended Capability structure. This field must return a Capability ID of 001Bh indicating that this is a PASID Extended Capability structure. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

7.8.9.2 PASID Capability Register (Offset 04h) §

§ Figure 7-188 details the allocation of register bits of the PASID Capability register; § Table 7-162 provides the respective bit definitions.

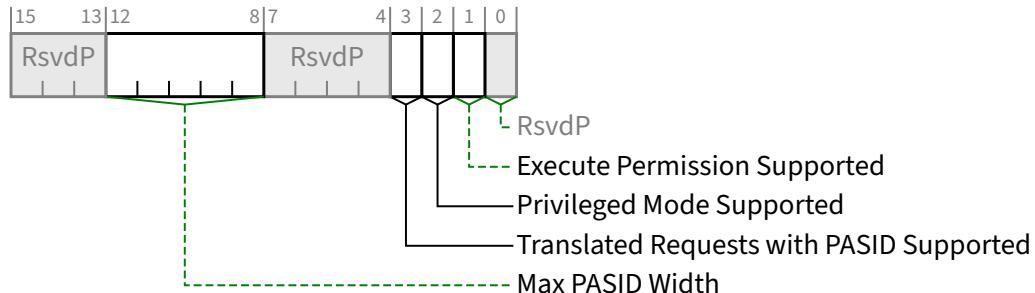


Figure 7-188 PASID Capability Register §

Table 7-162 PASID Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1 | Execute Permission Supported - If Set, an applicable Endpoint Function supports sending TLPs that have the <u>Execute Requested</u> bit Set. If Clear, the Endpoint Function will never Set the <u>Execute Requested</u> bit. It is strongly recommended that this bit be hardwired to 0b. | RO |
| 2 | Privileged Mode Supported - If Set, an applicable Endpoint Function supports operating in Privileged and Non-Privileged modes, and supports sending requests that have the <u>Privileged Mode Requested</u> bit Set. If Clear, the Endpoint Function will never Set the <u>Privileged Mode Requested</u> bit. | RO |
| 3 | Translated Requests with PASID Supported – If Set, indicates that an applicable Endpoint Function supports Translated Requests with PASID (see § Section 10.1.3). This bit is only permitted to be Set if the Endpoint Function supports ATS. | RO |
| 12:8 | Max PASID Width - Indicates the width of the PASID field supported by an applicable Endpoint Function. The value n indicates support for PASID values 0 through $2^n - 1$ (inclusive). The value 0 indicates support for a single PASID (0). The value 20 indicates support for all PASID values (20 bits). This field must be between 0 and 20 (inclusive). | RO |

7.8.9.3 PASID Control Register (Offset 06h) §

§ Figure 7-189 details the allocation of register bits of the PASID Control register; § Table 7-163 provides the respective bit definitions.

Base 6.4 vs Base 6.3

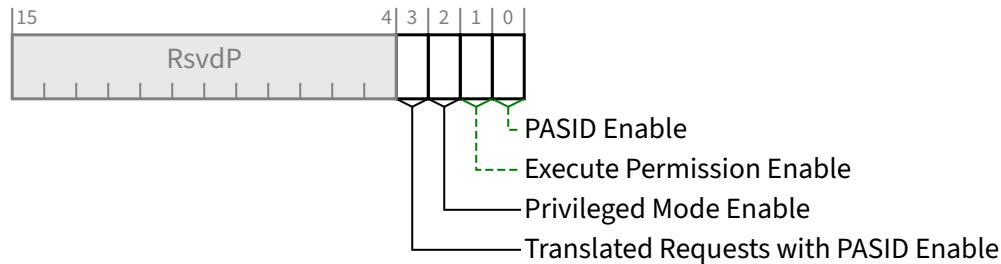


Figure 7-189 PASID Control Register §

Table 7-163 PASID Control Register §

| ↑↓Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|------------------------------------|--|---------------------------------|
| 0 | <p>PASID Enable - If Set, an applicable Endpoint Function is permitted to send and receive TLPs that contain a PASID TLP Prefix. If Clear, the Endpoint Function is not permitted to do so.</p> <p>Behavior is undefined if an applicable Endpoint Function supports ATS and this bit changes value when the Enable (E) bit in that Function's ATS Control Register is Set (see § Section 10.5.1.3).</p> <p>If Execute Permission Supported is Clear, this bit is RsvdP .</p> <p>Default is 0b.</p> | RW |
| 1 | <p>Execute Permission Enable - If Set, an applicable Endpoint Function Function is permitted to send Requests that have the <u>Execute Requested</u> bit Set. If Clear, the Endpoint is not permitted to do so.</p> <p>Behavior is undefined if an applicable Endpoint Function" supports ATS and this bit changes value when the Enable bit in that Function's ATS Control Register is Set (see § Section 10.5.1.3).</p> <p>If Execute Permission Supported is Clear, this bit is RsvdP .</p> <p>Default is 0b.</p> | RW / RsvdP (see description) |
| 2 | <p>Privileged Mode Enable - If Set, an applicable Endpoint Function is permitted to send Requests that have the <u>Privileged Mode Requested</u> bit Set. If Clear, the Endpoint Function is not permitted to do so.</p> <p>Behavior is undefined if an applicable Endpoint Function supports ATS and this bit changes value when the Enable bit in that Function's ATS Control Register is Set (see § Section 10.5.1.3).</p> <p>If Privileged Mode Supported is Clear, this bit is RsvdP .</p> <p>Default is 0b.</p> | RW / RsvdP (see description) |
| 3 | <p>Translated Requests with PASID Enable – When Set, the ATC associated with an applicable Endpoint Function is permitted to issue Translated Requests with a PASID TLP Prefix. If the ATC obtained a translation using a Translation Request with PASID, the corresponding Translated Request must carry a PASID that matches the PASID used to obtain the translation. Similarly, if the ATC obtained a translation using a Translation Request without a PASID, the corresponding Translated Request must not carry a PASID.</p> <p>When Clear, the ATC associated with the Endpoint Function is prohibited from issuing Translated Requests with a PASID.</p> <p>Behavior is undefined if an applicable Endpoint Function supports ATS this bit changes value when the Enable (E) bit in that Function's ATS Control Register is Set (see § Section 10.5.1.3).</p> | RW / RsvdP (see description) |

| Bit Location ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|--|--|------------|
| | If Translated Requests with PASID Supported bit is Clear, this bit is RsvdP . See (see § Section 10.1.3) for details. Default is 0b. | |

7.8.10 FRS Queueing Extended Capability §

The FRS Queueing Extended Capability is required for Root Ports and Root Complex Event Collectors that support the optional normative FRS Queueing capability. See § Section 6.22 . This extended capability is only permitted in Root Ports and Root Complex Event Collectors.

If this capability is present in a Function, that Function must also implement either MSI, MSI-X, or both.

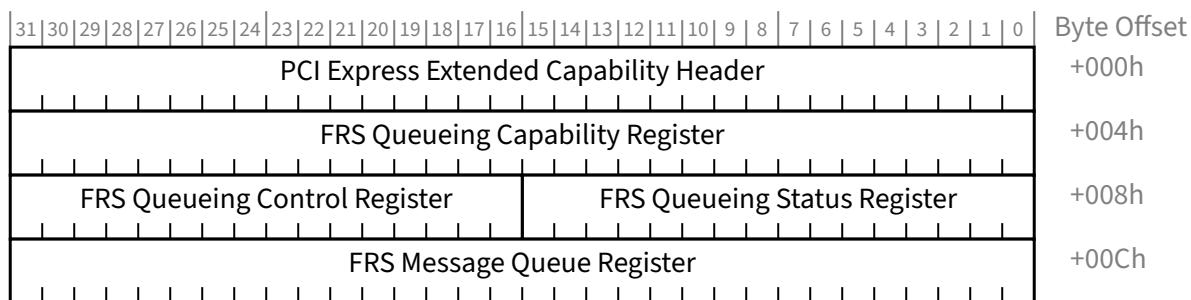


Figure 7-190 FRS Queueing Extended Capability §

7.8.10.1 FRS Queueing Extended Capability Header (Offset 00h) §

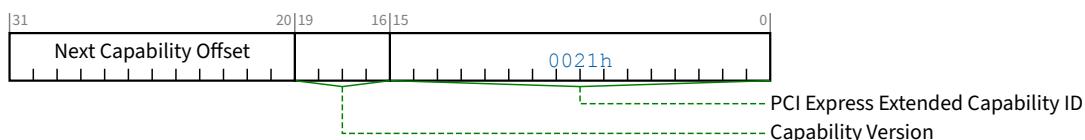


Figure 7-191 FRS Queueing Extended Capability Header §

Table 7-164 FRS Queueing Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the FRS Queueing Extended Capability is 0021h . | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

7.8.10.2 FRS Queueing Capability Register (Offset 04h) §

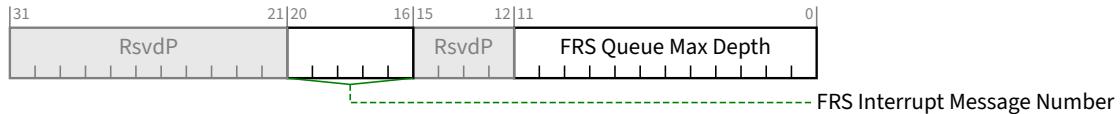


Figure 7-192 FRS Queueing Capability Register §

Table 7-165 FRS Queueing Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 11:0 | FRS Queue Max Depth - Indicates the implemented queue depth, with valid values ranging from 001h (queue depth of 1) to FFFh (queue depth of 4095) The value of <u>FRS Message Queue Depth</u> must not exceed this value. The value 000h is Reserved. | HwInit |
| 20:16 | FRS Interrupt Message Number - When MSI/MSI-X is implemented, this register indicates which MSI/MSI-X vector is used for the interrupt message generated in association with <u>FRS Message Received</u> or <u>FRS Message Overflow</u> . For MSI, the value in this register indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control Register for MSI. For MSI-X, the value in this register indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this register must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this register must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this register is undefined. | RO |

7.8.10.3 FRS Queueing Status Register (Offset 08h) §

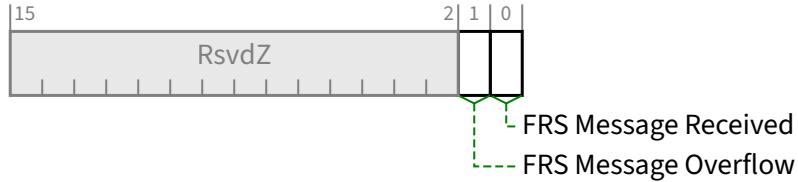


Figure 7-193 FRS Queueing Status Register §

Table 7-166 FRS Queueing Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>FRS Message Received - This bit is Set when a new FRS Message is Received or generated by this Root Port or Root Complex Event Collector.</p> <p>Root Ports must Clear this bit when the Link is <u>DL_Down</u>.</p> <p>Default value of this bit is 0b.</p> | RW1C |
| 1 | <p>FRS Message Overflow - This bit is Set if the FRS Message queue is full and a new FRS Message is received or generated by this Root Port or Root Complex Event Collector.</p> <p>Root Ports must Clear this bit when the Link is <u>DL_Down</u>.</p> <p>Default value of this bit is 0b.</p> | RW1C |

7.8.10.4 FRS Queueing Control Register (Offset 0Ah) §

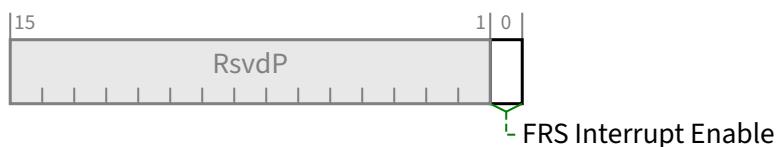


Figure 7-194 FRS Queueing Control Register §

Table 7-167 FRS Queueing Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>FRS Interrupt Enable - When Set and MSI or MSI-X is enabled, the Port must issue an MSI/MSI-X interrupt to indicate the 0b to 1b transition of either the <u>FRS Message Received</u> or the <u>FRS Message Overflow</u> bits.</p> <p>Default value of this bit is 0b.</p> | RW |

7.8.10.5 FRS Message Queue Register (Offset 0Ch) §

The FRS Message Queue Register contains fields from the oldest FRS message in the queue. It also indicates the number of FRS messages in the queue.

A write of any value that includes byte 0 to this register removes the oldest FRS Message from the queue and updates these fields. A write to this register when the queue is empty has no effect.

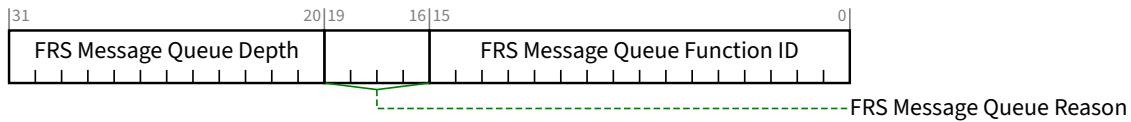


Figure 7-195 FRS Message Queue Register §

Table 7-168 FRS Message Queue Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | FRS Message Queue Function ID - Recorded from the Requester ID of the oldest FRS Message Received or generated by this Root Port or Root Complex Event Collector and still in the queue. Undefined if FRS Message Queue Depth is 000h. | RO |
| 19:16 | FRS Message Queue Reason - Recorded from the FRS Reason of the oldest FRS Message Received or generated by this Root Port or Root Complex Event Collector and still in the queue. Undefined if FRS Message Queue Depth is 000h. | RO |
| 31:20 | FRS Message Queue Depth - indicates the current number of FRS Messages in the queue. The value of 000h indicates an empty queue. Default value of this field is 000h. | RO |

7.8.11 Flattening Portal Bridge (FPB) Capability §

The Flattening Portal Bridge (FPB) Capability is an optional Capability that is required for any bridge Function that implements FPB. The FPB Capability structure is shown in § Figure 7-196 .

Base 6.4 vs Base 6.3

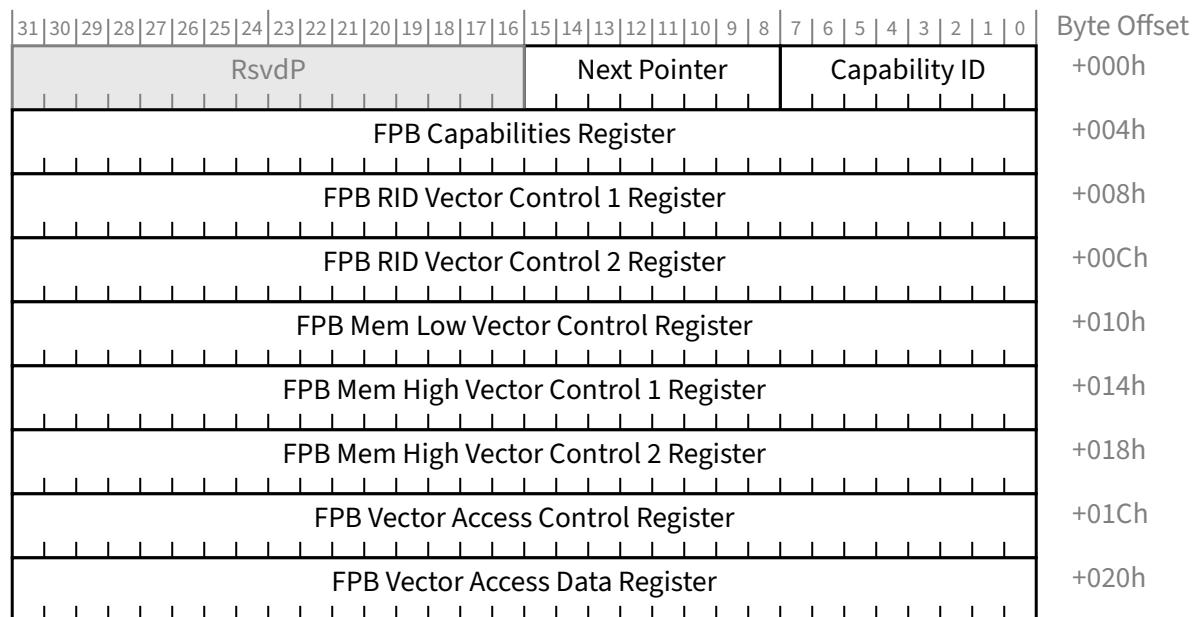


Figure 7-196 FPB Capability Structure §

If a Switch implements FPB then each of its Ports of the Switch must implement an FPB Capability Structure. A Root Complex is permitted to implement the FPB Capability Structure on some or on all of its Root Ports. A Root Complex is permitted to implement the FPB Capability for internal logical ↑↓busses.↓ ↑↓busses.↑

7.8.11.1 FPB Capability Header (Offset 00h) §

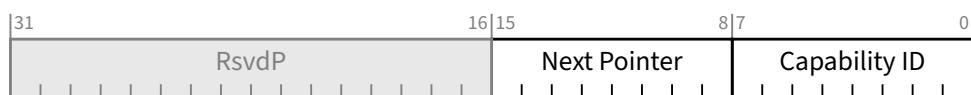


Figure 7-197 FPB Capability Header §

Table 7-169 FPB Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Capability ID - Must be set to 15h | RO |
| 15:8 | Next Pointer - Pointer to the next item in the capabilities list. Must be 00h for the final item in the list. | RO |

7.8.11.2 FPB Capabilities Register (Offset 04h) §

§ Figure 7-198 details allocation of register fields for FPB Capabilities register and § Table 7-170 describes the requirements for this register.

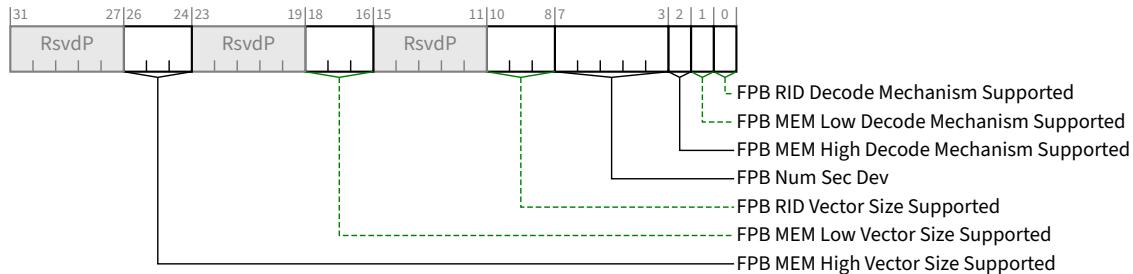


Figure 7-198 FPB Capabilities Register §

Table 7-170 FPB Capabilities Register §

| Bit Location | Register Description | Attributes | | | | | | | | | |
|--------------|---|------------------------------------|------|------------------------------------|------|----------|------------|------|----------|-------|--------|
| 0 | FPB RID Decode Mechanism Supported - If Set, indicates that the FPB RID Vector mechanism is supported. | HwInit | | | | | | | | | |
| 1 | FPB MEM Low Decode Mechanism Supported - If Set, indicates that the FPB MEM Low Vector mechanism is supported. | HwInit | | | | | | | | | |
| 2 | FPB MEM High Decode Mechanism Supported - If Set, indicates that the FPB Mem High mechanism is supported. | HwInit | | | | | | | | | |
| 7:3 | FPB Num Sec Dev - For Upstream Ports of Switches only, this field indicates the quantity of Device Numbers associated with the Secondary Side of the Upstream Port bridge. The quantity is determined by adding one to the numerical value of this field. Although it is recommended that Switch implementations assign Downstream Ports using all 8 allowed Functions per allocated Device Number, such that all Downstream Ports are assigned within a contiguous range of Device and Function Numbers, it is, however, explicitly permitted to assign Downstream Ports to Function Numbers that are not contiguous within the indicated range of Device Numbers, and system software is required to scan for Switch Downstream Ports at every Function Number within the indicated quantity of Device Numbers associated with the Secondary Side of the Upstream Port. This field is Reserved for Downstream Ports. | HwInit / RsvdP | | | | | | | | | |
| 10:8 | FPB RID Vector Size Supported - Indicates the size of the FPB RID Vector implemented in hardware, and constrains the allowed values software is permitted to write to the <u>FPB RID Vector Granularity</u> field. Defined encodings are: <table border="1"> <thead> <tr> <th>Value</th> <th>Size</th> <th>Allowed Granularities in RID units</th> </tr> </thead> <tbody> <tr> <td>000b</td> <td>256 bits</td> <td>8, 64, 256</td> </tr> <tr> <td>010b</td> <td>1 K bits</td> <td>8, 64</td> </tr> </tbody> </table> | Value | Size | Allowed Granularities in RID units | 000b | 256 bits | 8, 64, 256 | 010b | 1 K bits | 8, 64 | HwInit |
| Value | Size | Allowed Granularities in RID units | | | | | | | | | |
| 000b | 256 bits | 8, 64, 256 | | | | | | | | | |
| 010b | 1 K bits | 8, 64 | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | |
|--------------|---|-----------------------------------|------|-----------------------------------|----------|----------|----------------|------|----------|------------|----------|----------|----------|------|----------|--------|------|----------|---|--------|
| | <p>All other encodings are Reserved.</p> <p>If the FPB RID Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.</p> | | | | | | | | | | | | | | | | | | | |
| 18:16 | <p>FPB MEM Low Vector Size Supported - Indicates the size of the FPB MEM Low Vector implemented in hardware, and constrains the allowed values software is permitted to write to the <u>FPB MEM Low Vector Start</u> field.</p> <p>Defined encodings are:</p> <table border="1"> <thead> <tr> <th>Value</th><th>Size</th><th>Allowed Granularities in MB units</th></tr> </thead> <tbody> <tr> <td>000b</td><td>256 bits</td><td>1, 2, 4, 8, 16</td></tr> <tr> <td>001b</td><td>512 bits</td><td>1, 2, 4, 8</td></tr> <tr> <td>010b</td><td>1 K bits</td><td>1, 2, 4</td></tr> <tr> <td>011b</td><td>2 K bits</td><td>1, 2</td></tr> <tr> <td>100b</td><td>4 K bits</td><td>1</td></tr> </tbody> </table> <p>All other encodings are Reserved.</p> <p>If the FPB MEM Low Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.</p> | Value | Size | Allowed Granularities in MB units | 000b | 256 bits | 1, 2, 4, 8, 16 | 001b | 512 bits | 1, 2, 4, 8 | 010b | 1 K bits | 1, 2, 4 | 011b | 2 K bits | 1, 2 | 100b | 4 K bits | 1 | HwInit |
| Value | Size | Allowed Granularities in MB units | | | | | | | | | | | | | | | | | | |
| 000b | 256 bits | 1, 2, 4, 8, 16 | | | | | | | | | | | | | | | | | | |
| 001b | 512 bits | 1, 2, 4, 8 | | | | | | | | | | | | | | | | | | |
| 010b | 1 K bits | 1, 2, 4 | | | | | | | | | | | | | | | | | | |
| 011b | 2 K bits | 1, 2 | | | | | | | | | | | | | | | | | | |
| 100b | 4 K bits | 1 | | | | | | | | | | | | | | | | | | |
| 26:24 | <p>FPB MEM High Vector Size Supported - Indicates the size of the FPB MEM High Vector implemented in hardware.</p> <p>Defined encodings are:</p> <table border="1"> <thead> <tr> <th>Value</th><th>Size</th></tr> </thead> <tbody> <tr> <td>000b</td><td>256 bits</td></tr> <tr> <td>001b</td><td>512 bits</td></tr> <tr> <td>010b</td><td>1 K bits</td></tr> <tr> <td>011b</td><td>2 K bits</td></tr> <tr> <td>100b</td><td>4 K bits</td></tr> <tr> <td>101b</td><td>8 K bits</td></tr> </tbody> </table> <p>All other encodings are Reserved.</p> <p>All defined Granularities are allowed for all defined vector sizes.</p> <p>If the FPB MEM High Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.</p> | Value | Size | 000b | 256 bits | 001b | 512 bits | 010b | 1 K bits | 011b | 2 K bits | 100b | 4 K bits | 101b | 8 K bits | HwInit | | | | |
| Value | Size | | | | | | | | | | | | | | | | | | | |
| 000b | 256 bits | | | | | | | | | | | | | | | | | | | |
| 001b | 512 bits | | | | | | | | | | | | | | | | | | | |
| 010b | 1 K bits | | | | | | | | | | | | | | | | | | | |
| 011b | 2 K bits | | | | | | | | | | | | | | | | | | | |
| 100b | 4 K bits | | | | | | | | | | | | | | | | | | | |
| 101b | 8 K bits | | | | | | | | | | | | | | | | | | | |

7.8.11.3 FPB RID Vector Control 1 Register (Offset 08h) §

§ Figure 7-199 details allocation of register fields for FPB RID Control 1 register and § Table 7-174 describes the requirements for this register.

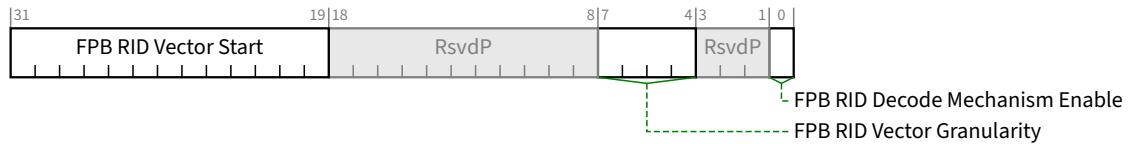


Figure 7-199 FPB RID Vector Control 1 Register §

Table 7-174 FPB RID Vector Control 1 Register §

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|--|------------|-------------|-------|--------|-------|---------|-------|----------|---------|
| 0 | <p>FPB RID Decode Mechanism Enable - When Set, enables the FPB RID Decode mechanism If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO , and in this case the value in this field is undefined. Default value of this bit is 0b.</p> | RW / RO | | | | | | | | |
| 7:4 | <p>FPB RID Vector Granularity - The value written by software to this field controls the granularity of the FPB RID Vector and the required alignment of the FPB RID Vector Start field (below). Defined encodings are:</p> <table border="1"> <thead> <tr> <th>Value</th><th>Granularity</th></tr> </thead> <tbody> <tr> <td>0000b</td><td>8 RIDs</td></tr> <tr> <td>0011b</td><td>64 RIDs</td></tr> <tr> <td>0101b</td><td>256 RIDs</td></tr> </tbody> </table> <p>All other encodings are Reserved. Based on the implemented FPB RID Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO , and the value in this field is undefined. For Downstream Ports, if the ARI Forwarding Enable bit in the Device Control 2 Register and the FPB RID Decode Mechanism Enable bit are Set, then software must program 0101b into this field, if this field is programmable. Default value for this field is 0000b.</p> | Value | Granularity | 0000b | 8 RIDs | 0011b | 64 RIDs | 0101b | 256 RIDs | RW / RO |
| Value | Granularity | | | | | | | | | |
| 0000b | 8 RIDs | | | | | | | | | |
| 0011b | 64 RIDs | | | | | | | | | |
| 0101b | 256 RIDs | | | | | | | | | |
| 31:19 | <p>FPB RID Vector Start - The value written by software to this field controls the offset at which the FPB RID Vector is applied. The value represents a RID offset in units of 8 RIDs, such that bit 0 of the FPB RID Vector represents the range of RIDs starting from the value represented in this register up to that value plus the FPB RID Vector</p> | RW / RO | | | | | | | | |

| Bit Location | Register Description | Attributes | | | | | | | | |
|----------------------------|--|----------------------------|----------------------------|-------|-----------------|-------|----------|-------|------------|--|
| | <p>Granularity minus 1, and bit 1 represents range from this register value plus granularity up to that value plus FPB RID Vector Granularity minus 1, etc.</p> <p>Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB RID Vector Granularity Field as indicated here:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>FPB RID Vector Granularity</th> <th>Start Alignment Constraint</th> </tr> </thead> <tbody> <tr> <td>0000b</td> <td><no constraint></td> </tr> <tr> <td>0011b</td> <td>...00 0b</td> </tr> <tr> <td>0101b</td> <td>...0000 0b</td> </tr> </tbody> </table> <p>All other encodings are Reserved.</p> <p>If this requirement is violated, the hardware behavior is undefined.</p> <p>For Downstream Ports, if the ARI Forwarding Enable bit in the Device Control 2 Register and the FPB RID Decode Mechanism Enable bit are Set, then software must program bits 23:19 of this field to a value of 0000 0b, and the hardware behavior is undefined if any other value is programmed.</p> <p>If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO , and the value in this field is undefined.</p> <p>Default value for this field is 0000 0000 0000 0b.</p> | FPB RID Vector Granularity | Start Alignment Constraint | 0000b | <no constraint> | 0011b | ...00 0b | 0101b | ...0000 0b | |
| FPB RID Vector Granularity | Start Alignment Constraint | | | | | | | | | |
| 0000b | <no constraint> | | | | | | | | | |
| 0011b | ...00 0b | | | | | | | | | |
| 0101b | ...0000 0b | | | | | | | | | |

7.8.11.4 FPB RID Vector Control 2 Register (Offset 0Ch) §

§ Figure 7-200 details allocation of register fields for FPB RID Vector Control 2 register and § Table 7-177 describes the requirements for this register

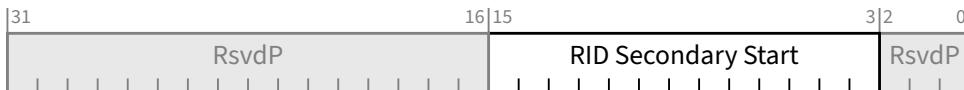


Figure 7-200 FPB RID Vector Control 2 Register §

Table 7-177 FPB RID Vector Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:3 | <p>RID Secondary Start - The value written by software to this field controls the RID offset at which Type 1 Configuration Requests passing downstream through the bridge must be converted to Type 0.</p> <p>Bits[2:0] of the RID offset are fixed by hardware as 000b and cannot be modified.</p> <p>For Downstream Ports, if the ARI Forwarding Enable bit in the Device Control 2 register is Set, then software must write bits 7:3 of this field to 0 0000b.</p> <p>If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO , and the value in this field is undefined.</p> <p>Default value for this field is 0000 0000 0000 0b.</p> | RW / RO |

7.8.11.5 FPB MEM Low Vector Control Register (Offset 10h) §

§ Figure 7-201 details allocation of register fields for FPB MEM Low Vector Control Register and § Table 7-178 describes the requirements for this register.

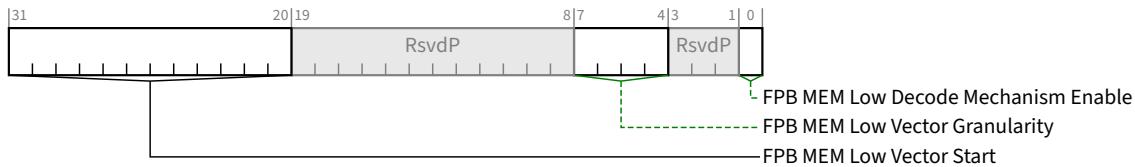


Figure 7-201 FPB MEM Low Vector Control Register §

Table 7-178 FPB MEM Low Vector Control Register §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | |
|--------------|--|------------|-------------|-------|------|-------|------|-------|------|-------|------|-------|-------|---------|
| 0 | <p>FPB MEM Low Decode Mechanism Enable - When Set, enables the FPB MEM Low Decode mechanism. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.</p> | RW / RO | | | | | | | | | | | | |
| 7:4 | <p>FPB MEM Low Vector Granularity - The value written by software to this field controls the granularity of the FPB MEM Low Vector, and the required alignment of the FPB MEM Low Vector Start field (below). Defined encodings are:</p> <table border="1"> <thead> <tr> <th>Value</th><th>Granularity</th></tr> </thead> <tbody> <tr> <td>0000b</td><td>1 MB</td></tr> <tr> <td>0001b</td><td>2 MB</td></tr> <tr> <td>0010b</td><td>4 MB</td></tr> <tr> <td>0011b</td><td>8 MB</td></tr> <tr> <td>0100b</td><td>16 MB</td></tr> </tbody> </table> <p>All other encodings are Reserved. Based on the implemented FPB MEM Low Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000b.</p> | Value | Granularity | 0000b | 1 MB | 0001b | 2 MB | 0010b | 4 MB | 0011b | 8 MB | 0100b | 16 MB | RW / RO |
| Value | Granularity | | | | | | | | | | | | | |
| 0000b | 1 MB | | | | | | | | | | | | | |
| 0001b | 2 MB | | | | | | | | | | | | | |
| 0010b | 4 MB | | | | | | | | | | | | | |
| 0011b | 8 MB | | | | | | | | | | | | | |
| 0100b | 16 MB | | | | | | | | | | | | | |
| 31:20 | <p>FPB MEM Low Vector Start - The value written by software to this field sets bits 31:20 of the base address at which the FPB MEM Low Vector is applied. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM Low Vector Granularity field as indicated here:</p> | RW / RO | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | |
|--------------------------------|---|--------------------------------|--|------------|-------|--|-----------------|-------|--|-------|-------|--|--------|-------|--|---------|-------|--|----------|--|
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">FPB MEM Low Vector Granularity</th> <th>Constraint</th> </tr> </thead> <tbody> <tr> <td>0000b</td> <td></td> <td><no constraint></td> </tr> <tr> <td>0001b</td> <td></td> <td>...0b</td> </tr> <tr> <td>0010b</td> <td></td> <td>...00b</td> </tr> <tr> <td>0011b</td> <td></td> <td>...000b</td> </tr> <tr> <td>0100b</td> <td></td> <td>...0000b</td> </tr> </tbody> </table> <p>If this requirement is violated, the hardware behavior is undefined.</p> <p>If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO , and the value in this field is undefined.</p> <p>Default value for this field is 000h.</p> | FPB MEM Low Vector Granularity | | Constraint | 0000b | | <no constraint> | 0001b | | ...0b | 0010b | | ...00b | 0011b | | ...000b | 0100b | | ...0000b | |
| FPB MEM Low Vector Granularity | | Constraint | | | | | | | | | | | | | | | | | | |
| 0000b | | <no constraint> | | | | | | | | | | | | | | | | | | |
| 0001b | | ...0b | | | | | | | | | | | | | | | | | | |
| 0010b | | ...00b | | | | | | | | | | | | | | | | | | |
| 0011b | | ...000b | | | | | | | | | | | | | | | | | | |
| 0100b | | ...0000b | | | | | | | | | | | | | | | | | | |

7.8.11.6 FPB MEM High Vector Control 1 Register (Offset 14h) §

§ Figure 7-202 details allocation of register fields for FPB MEM High Vector Control 1 Register and § Table 7-181 describes the requirements for this register.

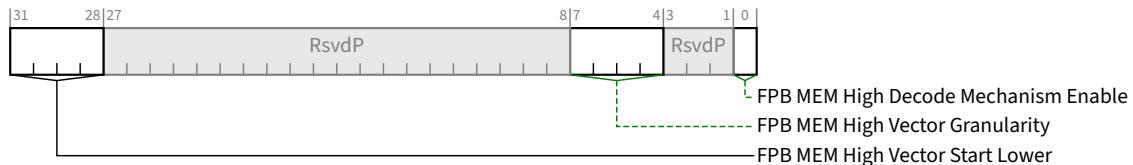


Figure 7-202 FPB MEM High Vector Control 1 Register §

Table 7-181 FPB MEM High Vector Control 1 Register §

| Bit Location | Register Description | Attributes | | | | |
|--------------|--|------------|-------------|-------|--------|---------|
| 0 | <p>FPB MEM High Decode Mechanism Enable - When Set, enables the FPB MEM High Decode mechanism.</p> <p>If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO , and in this case the value in this field is undefined.</p> <p>Default value of this bit is 0b.</p> | RW / RO | | | | |
| 7:4 | <p>FPB MEM High Vector Granularity - The value written by software to this field controls the granularity of the FPB MEM High Vector, and the required alignment of the FPB MEM High Vector Start Lower field (below).</p> <p>Software is permitted to select any allowed Granularity from the table below regardless of the value in the FPB MEM High Vector Size Supported field.</p> <p>Defined encodings are:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Value</th> <th>Granularity</th> </tr> </thead> <tbody> <tr> <td>0000b</td> <td>256 MB</td> </tr> </tbody> </table> | Value | Granularity | 0000b | 256 MB | RW / RO |
| Value | Granularity | | | | | |
| 0000b | 256 MB | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | |
|---------------------------------|--|---------------------------------|-------------|-------|-----------------|-------|-------|-------|--------|-------|---------|-------|----------|-------|------------|-------|-------------|-------|--------------|---------|
| | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Value</th><th>Granularity</th></tr> </thead> <tbody> <tr><td>0001b</td><td>512 MB</td></tr> <tr><td>0010b</td><td>1 GB</td></tr> <tr><td>0011b</td><td>2 GB</td></tr> <tr><td>0100b</td><td>4 GB</td></tr> <tr><td>0101b</td><td>8 GB</td></tr> <tr><td>0110b</td><td>16 GB</td></tr> <tr><td>0111b</td><td>32 GB</td></tr> </tbody> </table> <p>All other encodings are Reserved.</p> <p>Based on the implemented FPB MEM High Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0.</p> <p>If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined.</p> <p>Default value for this field is 0000b.</p> | Value | Granularity | 0001b | 512 MB | 0010b | 1 GB | 0011b | 2 GB | 0100b | 4 GB | 0101b | 8 GB | 0110b | 16 GB | 0111b | 32 GB | | | |
| Value | Granularity | | | | | | | | | | | | | | | | | | | |
| 0001b | 512 MB | | | | | | | | | | | | | | | | | | | |
| 0010b | 1 GB | | | | | | | | | | | | | | | | | | | |
| 0011b | 2 GB | | | | | | | | | | | | | | | | | | | |
| 0100b | 4 GB | | | | | | | | | | | | | | | | | | | |
| 0101b | 8 GB | | | | | | | | | | | | | | | | | | | |
| 0110b | 16 GB | | | | | | | | | | | | | | | | | | | |
| 0111b | 32 GB | | | | | | | | | | | | | | | | | | | |
| 31:28 | <p>FPB MEM High Vector Start Lower - The value written by software to this field sets the lower bits of the base address at which the FPB MEM High Vector is applied.</p> <p>Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM High Vector Granularity Field as indicated here:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>FPB MEM High Vector Granularity</th><th>Constraint</th></tr> </thead> <tbody> <tr><td>0000b</td><td><no constraint></td></tr> <tr><td>0001b</td><td>...0b</td></tr> <tr><td>0010b</td><td>...00b</td></tr> <tr><td>0011b</td><td>...000b</td></tr> <tr><td>0100b</td><td>...0000b</td></tr> <tr><td>0101b</td><td>...0 0000b</td></tr> <tr><td>0110b</td><td>...00 0000b</td></tr> <tr><td>0111b</td><td>...000 0000b</td></tr> </tbody> </table> <p>If this requirement is violated, the hardware behavior is undefined.</p> <p>If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined.</p> <p>Default value for this field is 0h.</p> | FPB MEM High Vector Granularity | Constraint | 0000b | <no constraint> | 0001b | ...0b | 0010b | ...00b | 0011b | ...000b | 0100b | ...0000b | 0101b | ...0 0000b | 0110b | ...00 0000b | 0111b | ...000 0000b | RW / RO |
| FPB MEM High Vector Granularity | Constraint | | | | | | | | | | | | | | | | | | | |
| 0000b | <no constraint> | | | | | | | | | | | | | | | | | | | |
| 0001b | ...0b | | | | | | | | | | | | | | | | | | | |
| 0010b | ...00b | | | | | | | | | | | | | | | | | | | |
| 0011b | ...000b | | | | | | | | | | | | | | | | | | | |
| 0100b | ...0000b | | | | | | | | | | | | | | | | | | | |
| 0101b | ...0 0000b | | | | | | | | | | | | | | | | | | | |
| 0110b | ...00 0000b | | | | | | | | | | | | | | | | | | | |
| 0111b | ...000 0000b | | | | | | | | | | | | | | | | | | | |

7.8.11.7 FPB MEM High Vector Control 2 Register (Offset 18h) §

§ Figure 7-203 details allocation of register fields for FPB MEM High Vector Control 2 Register and § Table 7-184 describes the requirements for this register.

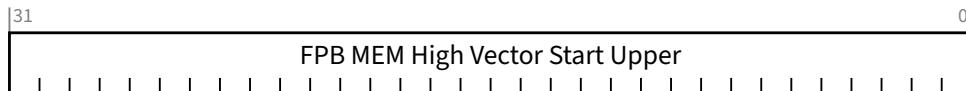


Figure 7-203 FPB MEM High Vector Control 2 Register §

Table 7-184 FPB MEM High Vector Control 2 Register §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | |
|---------------------------------|--|---------------------------------|------------|-------|-----------------|-------|-----------------|-------|-----------------|-------|-----------------|-------|-----------------|-------|-------|-------|--------|-------|---------|---------|
| 31:0 | <p>FPB MEM High Vector Start Upper - The value written by software to this field sets bits 63:32 of the base address at which the FPB MEM High Vector is applied.</p> <p>Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM High Vector Granularity Field as indicated here:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>FPB MEM High Vector Granularity</th><th>Constraint</th></tr> </thead> <tbody> <tr><td>0000b</td><td><no constraint></td></tr> <tr><td>0001b</td><td><no constraint></td></tr> <tr><td>0010b</td><td><no constraint></td></tr> <tr><td>0011b</td><td><no constraint></td></tr> <tr><td>0100b</td><td><no constraint></td></tr> <tr><td>0101b</td><td>...0b</td></tr> <tr><td>0110b</td><td>...00b</td></tr> <tr><td>0111b</td><td>...000b</td></tr> </tbody> </table> <p>If this requirement is violated, the hardware behavior is undefined</p> <p>If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined.</p> <p>Default value for this field is 0000 0000h.</p> | FPB MEM High Vector Granularity | Constraint | 0000b | <no constraint> | 0001b | <no constraint> | 0010b | <no constraint> | 0011b | <no constraint> | 0100b | <no constraint> | 0101b | ...0b | 0110b | ...00b | 0111b | ...000b | RW / RO |
| FPB MEM High Vector Granularity | Constraint | | | | | | | | | | | | | | | | | | | |
| 0000b | <no constraint> | | | | | | | | | | | | | | | | | | | |
| 0001b | <no constraint> | | | | | | | | | | | | | | | | | | | |
| 0010b | <no constraint> | | | | | | | | | | | | | | | | | | | |
| 0011b | <no constraint> | | | | | | | | | | | | | | | | | | | |
| 0100b | <no constraint> | | | | | | | | | | | | | | | | | | | |
| 0101b | ...0b | | | | | | | | | | | | | | | | | | | |
| 0110b | ...00b | | | | | | | | | | | | | | | | | | | |
| 0111b | ...000b | | | | | | | | | | | | | | | | | | | |

7.8.11.8 FPB Vector Access Control Register (Offset 1Ch) §

§ Figure 7-204 details allocation of register fields for FPB Vector Access Control register and § Table 7-186 describes the requirements for this register.

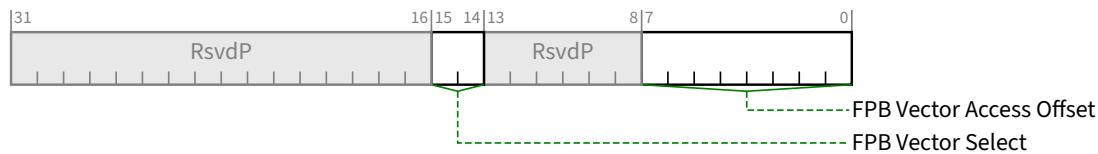


Figure 7-204 FPB Vector Access Control Register §

Table 7-186 FPB Vector Access Control Register §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | | | | | | |
|-----------------------|---|-----------------------|-------------|----------------------|------|-----|------------------|------|-----|------------------|------|-----|------------------|------|-----|------------------|------|-----|----------------|------|-----|-----|---------|
| 7:0 | <p>FPB Vector Access Offset - The value in this field indicates the offset of the DWORD portion of the FPB RID, MEM Low or MEM High, Vector that can be read or written by means of the <u>FPB Vector Access Data register</u>.</p> <p>The selection of RID, MEM Low or MEM High is made by the value written to the <u>FPB Vector Select</u> field.</p> <p>The bits of this field map to the offset according to the value in the corresponding FPB RID, MEM Low, or MEM High Vector Size Supported field as shown here:</p> <table border="1"> <thead> <tr> <th>Vector Size Supported</th><th>Offset Bits</th><th>Vector Access Offset</th></tr> </thead> <tbody> <tr> <td>000b</td><td>2:0</td><td>2:0 (7:3 unused)</td></tr> <tr> <td>001b</td><td>3:0</td><td>3:0 (7:4 unused)</td></tr> <tr> <td>010b</td><td>4:0</td><td>4:0 (7:5 unused)</td></tr> <tr> <td>011b</td><td>5:0</td><td>5:0 (7:6 unused)</td></tr> <tr> <td>100b</td><td>6:0</td><td>6:0 (7 unused)</td></tr> <tr> <td>101b</td><td>7:0</td><td>7:0</td></tr> </tbody> </table> <p>All other encodings are Reserved.</p> <p>Bits in this field that are unused per the table above must be written by software as 0b, and are permitted but not required to be implemented as RO .</p> <p>Default value for this field is 00h</p> | Vector Size Supported | Offset Bits | Vector Access Offset | 000b | 2:0 | 2:0 (7:3 unused) | 001b | 3:0 | 3:0 (7:4 unused) | 010b | 4:0 | 4:0 (7:5 unused) | 011b | 5:0 | 5:0 (7:6 unused) | 100b | 6:0 | 6:0 (7 unused) | 101b | 7:0 | 7:0 | RW / RO |
| Vector Size Supported | Offset Bits | Vector Access Offset | | | | | | | | | | | | | | | | | | | | | |
| 000b | 2:0 | 2:0 (7:3 unused) | | | | | | | | | | | | | | | | | | | | | |
| 001b | 3:0 | 3:0 (7:4 unused) | | | | | | | | | | | | | | | | | | | | | |
| 010b | 4:0 | 4:0 (7:5 unused) | | | | | | | | | | | | | | | | | | | | | |
| 011b | 5:0 | 5:0 (7:6 unused) | | | | | | | | | | | | | | | | | | | | | |
| 100b | 6:0 | 6:0 (7 unused) | | | | | | | | | | | | | | | | | | | | | |
| 101b | 7:0 | 7:0 | | | | | | | | | | | | | | | | | | | | | |
| 15:14 | <p>FPB Vector Select - The value written to this field selects the Vector to be accessed at the indicated FPB Vector Access Offset . Software must only write this field with values that correspond to supported FPB mechanisms, otherwise the results are undefined.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b RID 01b MEM Low 10b MEM High 11b Reserved <p>Default value for this field is 00b</p> | RW | | | | | | | | | | | | | | | | | | | | | |

7.8.11.9 FPB Vector Access Data Register (Offset 20h) §

§ Figure 7-205 details allocation of register fields for FPB Vector Access Data Register and § Table 7-188 describes the requirements for this register.

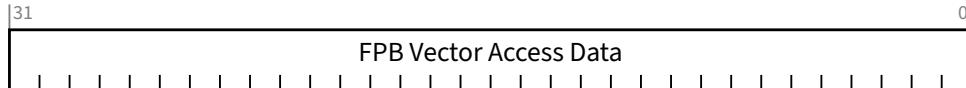


Figure 7-205 FPB Vector Access Data Register §

Table 7-188 FPB Vector Access Data Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:0 | <p>FPB Vector Access Data - Reads from this register return the DW of data from the FPB Vector at the location determined by the value in the FPB Vector Access Offset Register. Writes to this register replace the DW of data from the FPB Vector at the location determined by the value in the FPB Vector Access Offset Register.</p> <p>Behavior of this field is undefined if software programs unsupported values for FPB Vector Select or FPB Vector Access Offset fields, however hardware is required to complete the access to this register normally.</p> <p>Default value for this field is 0000 0000h</p> | RW |

7.8.12 Flit Performance Measurement Extended Capability §

This capability is optional. This capability is permitted in Downstream Ports, in Function 0 of an Upstream Port, and in RCRBs. This capability is not permitted in other Functions.

This capability is only used in Flit Mode. The capability has no effect in Non-Flit Mode.

The registers LTSSM Performance Measurement Status 1 Register through LTSSM Performance Measurement Status 5 Register are optional. The number implemented is contained in LTSSM Tracking Register Count. Unimplemented registers do not exist (i.e., the capability becomes shorter than shown in § Figure 7-206)

§ Figure 7-206 details allocation of the register bits in the Flit Performance Measurement Extended Capability structure.

Base 6.4 vs Base 6.3

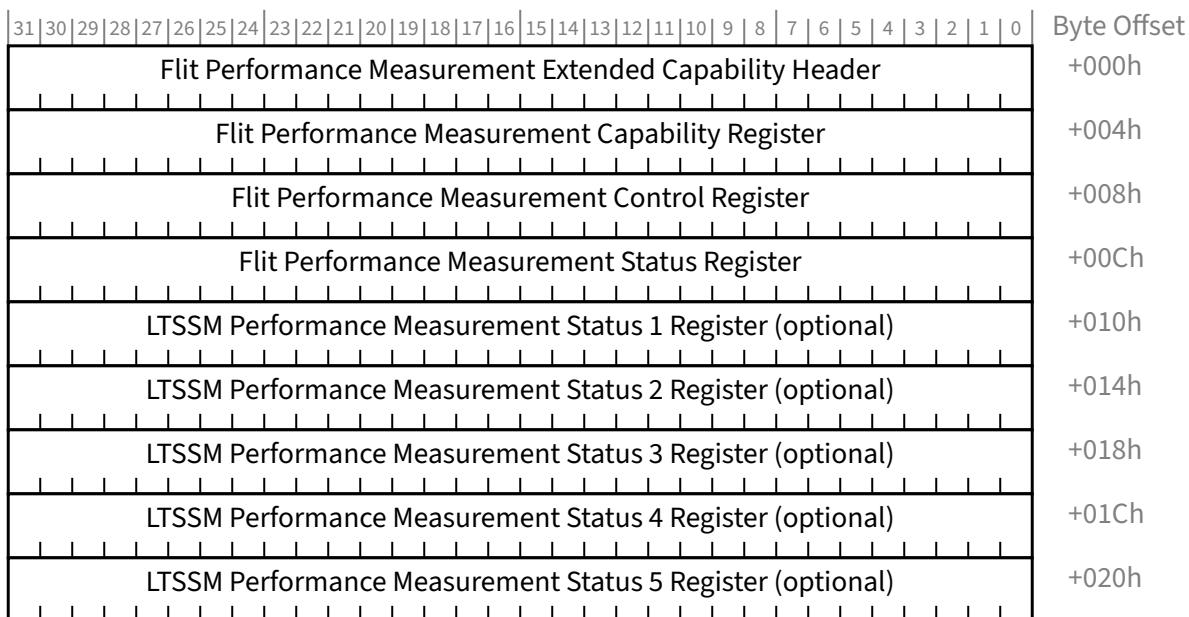


Figure 7-206 Flit Performance Measurement Extended Capability Structure §

7.8.12.1 Flit Performance Measurement Extended Capability Header (Offset 00h) §

§ Figure 7-207 details allocation of the register fields in the Flit Performance Measurement Extended Capability Header ;
 § Table 7-189 provides the respective bit definitions.

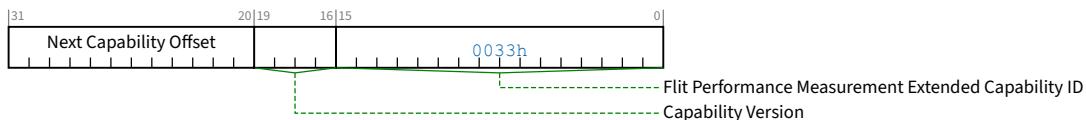


Figure 7-207 Flit Performance Measurement Extended Capability Header §

Table 7-189 Flit Performance Measurement Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | Flit Performance Measurement Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Flit Performance Measurement Extended Capability is 0033h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:20 | <p>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.</p> <p>For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.</p> | RO |

7.8.12.2 Flit Performance Measurement Capability Register (Offset 04h) §

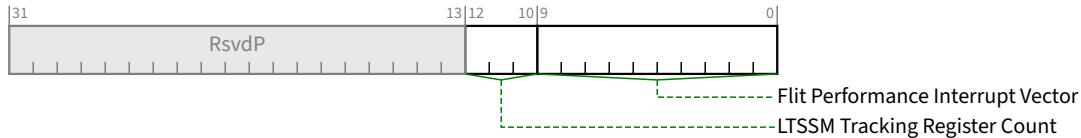


Figure 7-208 Flit Performance Measurement Capability Register §

Table 7-190 Flit Performance Measurement Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 9:0 | <p>Flit Performance Interrupt Vector – contains the MSI or MSI-X Vector number used by this mechanism. If both MSI and MSI-X are implemented, this field is permitted to change value based on which one is enabled. Additionally, when MSI is enabled, this field is permitted to change value based on the value of Multiple Message Enable</p> | RO |
| 12:10 | <p>LTSSM Tracking Register Count – Indicates the number of simultaneous LTSSM tracking events that are supported. Value must be between 0 and 5.</p> | HwInit |

7.8.12.3 Flit Performance Measurement Control Register (Offset 08h) §

The status register in capability indicates how many events can be simultaneously tracked for the LTSSM state transition tracker. Software must ensure that it does not enable more bits than the Port can enable.

Behavior is undefined if bits 31:1 of this register are changed while Flit Latency Measurement is running (Flit Latency Measurement Enable is 1b and either Flit Response Type is non-zero and Flit Latency Tracking Status is 00b or 01b, or LTSSM State Transition Tracker is non-zero and any of the enabled LTSSM State Transition Tracking Status fields are 00b or 01b).

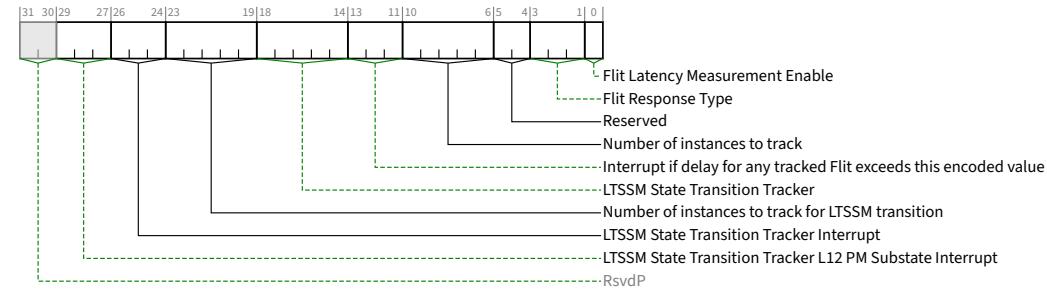


Figure 7-209 Flit Performance Measurement Control Register §

Table 7-191 Flit Performance Measurement Control Register §

| Bit Location | Register Description | Attributes | | | | | | | | |
|----------------|--|----------------|---|---------------|---|-------------|---|---------------|----------|----|
| 0 | <p>Flit Latency Measurement Enable – Setting this bit to 1b enables and starts measuring the Ack/ Nak/ Replay latency of a Flit. Writing a 0b to this bit when a measurement is in progress stops the measurement and sets Flit Latency Tracking Status to 10b. Writing this bit to 1b when it is already 1b has no effect.</p> <p>Unit of measurement is 8 ns.</p> <p>Default is Zero .</p> | RW | | | | | | | | |
| 3:1 | <p>Flit Response Type – Setting the associated bit to 1b enables measuring the Nak to Replay, Flit to Nak, or Flit to Ack latency of a Flit, depending on which bit was written. Behavior is undefined if this field changes value while a measurement is in progress. Behavior is undefined if this field contains a Reserved encoding and Flit Latency Measurement Enable is 1b.</p> <table> <tr> <td>001b</td><td>Flit to Ack Latency – this measures the time period from sending an original Flit to receiving an Ack for precisely that Flit at the same Link Width. It does not include Flits that were replayed or Flits that were implicitly Ack'ed by receiving the Ack for a subsequent Flit.</td></tr> <tr> <td>010b</td><td>Flit to Nak Latency – this measures the time period from sending an original Flit to receiving an Nak for precisely that Flit at the same Link Width. It does not include Flits that were replayed or Flits that were implicitly Nak'ed by receiving a Nak for an earlier Flit.</td></tr> <tr> <td>100b</td><td>Nak to Replay Latency – this measures the time period from sending the first Flit containing a Nak for a given sequence number to receiving the first replay of the requested Flit at the same Link Width</td></tr> <tr> <td>Others</td><td>Reserved</td></tr> </table> <p>Default is Zero .</p> | 001b | Flit to Ack Latency – this measures the time period from sending an original Flit to receiving an Ack for precisely that Flit at the same Link Width. It does not include Flits that were replayed or Flits that were implicitly Ack'ed by receiving the Ack for a subsequent Flit. | 010b | Flit to Nak Latency – this measures the time period from sending an original Flit to receiving an Nak for precisely that Flit at the same Link Width. It does not include Flits that were replayed or Flits that were implicitly Nak'ed by receiving a Nak for an earlier Flit. | 100b | Nak to Replay Latency – this measures the time period from sending the first Flit containing a Nak for a given sequence number to receiving the first replay of the requested Flit at the same Link Width | Others | Reserved | RW |
| 001b | Flit to Ack Latency – this measures the time period from sending an original Flit to receiving an Ack for precisely that Flit at the same Link Width. It does not include Flits that were replayed or Flits that were implicitly Ack'ed by receiving the Ack for a subsequent Flit. | | | | | | | | | |
| 010b | Flit to Nak Latency – this measures the time period from sending an original Flit to receiving an Nak for precisely that Flit at the same Link Width. It does not include Flits that were replayed or Flits that were implicitly Nak'ed by receiving a Nak for an earlier Flit. | | | | | | | | | |
| 100b | Nak to Replay Latency – this measures the time period from sending the first Flit containing a Nak for a given sequence number to receiving the first replay of the requested Flit at the same Link Width | | | | | | | | | |
| Others | Reserved | | | | | | | | | |
| 5:4 | <p>Reserved – This field is permitted to be implemented as either RsvdP or RW with no effects. If implemented as RW , default is Zero .</p> | RsvdP / RW | | | | | | | | |
| 10:6 | <p>Number of instances to track</p> <table> <tr> <td>0 0000b</td><td>track the worst-case delay</td></tr> <tr> <td>Others</td><td>track cumulative delay of the indicated number of Flits</td></tr> </table> <p>Behavior is undefined if this field changes value while a measurement is in progress.</p> | 0 0000b | track the worst-case delay | Others | track cumulative delay of the indicated number of Flits | RW | | | | |
| 0 0000b | track the worst-case delay | | | | | | | | | |
| Others | track cumulative delay of the indicated number of Flits | | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | |
|----------------|---|----------------|---|---------------|---|---------------|--------------|---------------|--------------|---------------|--------------|----|
| | <p>When this field is <u>Zero</u>, measurement completes when Flit Latency Measurement Enable is cleared. When this field is <u>non-Zero</u>, measurement completes when the indicated number of Flits have been tracked. Default is <u>Zero</u>.</p> | | | | | | | | | | | |
| 13:11 | <p>Interrupt if delay for any tracked Flit exceeds this encoded value</p> <table> <tr> <td>000b</td><td>000b – do not generate an interrupt</td></tr> <tr> <td>001b</td><td>100 ns</td></tr> <tr> <td>010b</td><td>200 ns</td></tr> <tr> <td>011b</td><td>300 ns</td></tr> <tr> <td>Others</td><td>Reserved</td></tr> </table> <p>Behavior is undefined if this field changes value while a measurement is in progress. Default is <u>Zero</u>.</p> | 000b | 000b – do not generate an interrupt | 001b | 100 ns | 010b | 200 ns | 011b | 300 ns | Others | Reserved | RW |
| 000b | 000b – do not generate an interrupt | | | | | | | | | | | |
| 001b | 100 ns | | | | | | | | | | | |
| 010b | 200 ns | | | | | | | | | | | |
| 011b | 300 ns | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | |
| 18:14 | <p>LTSSM State Transition Tracker – Each bit counts as one independent event:</p> <table> <tr> <td>Bit 14</td><td>L0 to Recovery due to a Framing Error / software directed while in L0 .</td></tr> <tr> <td>Bit 15</td><td>L0p – Electrical Idle to start of Data Stream on Lane on an upconfig.</td></tr> <tr> <td>Bit 16</td><td>L1.0 to L0 .</td></tr> <tr> <td>Bit 17</td><td>L1.1 to L0 .</td></tr> <tr> <td>Bit 18</td><td>L1.2 to L0 .</td></tr> </table> <p>Behavior is undefined if the number of bits set in this field is greater than <u>LTSSM Tracking Register Count</u>. Default is <u>Zero</u>.</p> | Bit 14 | L0 to Recovery due to a Framing Error / software directed while in L0 . | Bit 15 | L0p – Electrical Idle to start of Data Stream on Lane on an upconfig. | Bit 16 | L1.0 to L0 . | Bit 17 | L1.1 to L0 . | Bit 18 | L1.2 to L0 . | RW |
| Bit 14 | L0 to Recovery due to a Framing Error / software directed while in L0 . | | | | | | | | | | | |
| Bit 15 | L0p – Electrical Idle to start of Data Stream on Lane on an upconfig. | | | | | | | | | | | |
| Bit 16 | L1.0 to L0 . | | | | | | | | | | | |
| Bit 17 | L1.1 to L0 . | | | | | | | | | | | |
| Bit 18 | L1.2 to L0 . | | | | | | | | | | | |
| 23:19 | <p>Number of instances to track for LTSSM transition</p> <table> <tr> <td>0 0000b</td><td>track the worst-case delay</td></tr> <tr> <td>Others</td><td>aggregate delay of the number presented here</td></tr> </table> <p>Default is <u>Zero</u>.</p> | 0 0000b | track the worst-case delay | Others | aggregate delay of the number presented here | RW | | | | | | |
| 0 0000b | track the worst-case delay | | | | | | | | | | | |
| Others | aggregate delay of the number presented here | | | | | | | | | | | |
| 26:24 | <p>LTSSM State Transition Tracker Interrupt - Interrupt if any of the events covered by the low 3 bits of LTSSM State Transition Tracker (bits 16:14 of this register) exceeds this encoded value:</p> <table> <tr> <td>000b</td><td>no interrupt generated</td></tr> <tr> <td>001b</td><td>6.4 ms</td></tr> <tr> <td>010b</td><td>12.8 ms</td></tr> <tr> <td>011b</td><td>19.2 ms</td></tr> <tr> <td>Others</td><td>Reserved</td></tr> </table> <p>Default is <u>Zero</u>.</p> | 000b | no interrupt generated | 001b | 6.4 ms | 010b | 12.8 ms | 011b | 19.2 ms | Others | Reserved | RW |
| 000b | no interrupt generated | | | | | | | | | | | |
| 001b | 6.4 ms | | | | | | | | | | | |
| 010b | 12.8 ms | | | | | | | | | | | |
| 011b | 19.2 ms | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | |
| 29:27 | <p>LTSSM State Transition Tracker L12 PM Substate Interrupt - Interrupt if any of the events covered by upper 2 bits of LTSSM State Transition Tracker (bits 18:17 of this register) exceeds this value:</p> <table> <tr> <td>000b</td><td>no interrupt generated</td></tr> <tr> <td>001b</td><td>1 sec</td></tr> <tr> <td>010b</td><td>2 sec</td></tr> <tr> <td>011b</td><td>3 sec</td></tr> <tr> <td>100b</td><td>4 sec</td></tr> </table> | 000b | no interrupt generated | 001b | 1 sec | 010b | 2 sec | 011b | 3 sec | 100b | 4 sec | RW |
| 000b | no interrupt generated | | | | | | | | | | | |
| 001b | 1 sec | | | | | | | | | | | |
| 010b | 2 sec | | | | | | | | | | | |
| 011b | 3 sec | | | | | | | | | | | |
| 100b | 4 sec | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>101b 5 sec</p> <p>110b 10 sec</p> <p>111b Reserved</p> <p>Default is Zero .</p> | |

7.8.12.4 Flit Performance Measurement Status Register (Offset 0Ch) §

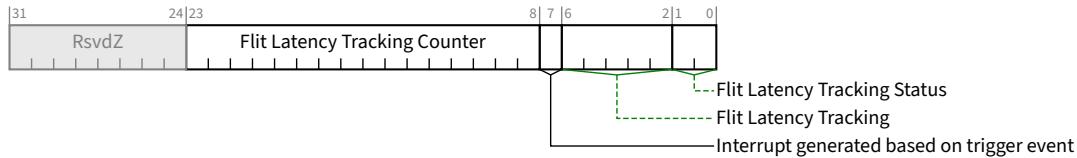


Figure 7-210 Flit Performance Measurement Status Register §

Table 7-192 Flit Performance Measurement Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1:0 | <p>Flit Latency Tracking Status</p> <p>00b Not started</p> <p>01b Started</p> <p>10b Completed</p> <p>11b Completed with Error (counter overflow)</p> <p>Default is Zero .</p> | RO |
| 6:2 | <p>Flit Latency Tracking – This field indicates the exact number of Flits which have been tracked and recorded in Flit Latency Tracking Counter . This field does not roll over.</p> <p>When Number of instances to track is non-Zero, this field will be less than or equal to Number of instances to track .</p> <p>When Number of instances to track is Zero , this field may contain any value.</p> <p>If Flit Latency Tracking Status is 11b, this field is undefined.</p> <p>Default is Zero .</p> | RO |
| 7 | <p>Interrupt generated based on trigger event – this bit is Set to 1b if an interrupt is generated based on the trigger event count due to Flit Latency Tracking. While this bit is Set to 1b, no new interrupts will be generated based on the trigger event.</p> <p>Default is Zero .</p> | RW1C |
| 23:8 | <p>Flit Latency Tracking Counter</p> <p>If Number of instances to track is non-Zero, this field contains the sum of the latency values measures for the tracked Flits. Software can divide this value by the Flit Latency Tracking field to compute the average latency.</p> <p>If Number of instances to track is Zero , this field contains the largest latency measured for the tracked Flits.</p> <p>If Flit Latency Tracking Status is 11b, this field is undefined.</p> | RO |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | The measurement unit is 8 ns ²⁰¹ . Default is Zero . | |

IMPLEMENTATION NOTE: FLIT PERFORMANCE MEASUREMENT OPERATION §

Flit Performance Measurement allows software to measure the Link Latency for the selected Flit types. These measurements reflect the timer period from when the tracked Flit is sent to when the tracking complete Flit is received. It is strongly recommended that these two events be consistent with each other (e.g., measure from when the first bit of a Flit was transmitted to when the first bit of a Flit was received). When performing a measurement, software should disable Link width changes by configuring Target Link Width and Hardware Autonomous Width Disable. Not doing this can result in inaccurate measurement values.

7.8.12.5 LTSSM Performance Measurement Status Register (Offsets 10h to 20h) §

Up to 5 instances of the following register are supported. Each register instance supports measurement of one LTSSM state transition tracking. If multiple entries are supported the order of association is based on the bits being enabled in the control register. Software must not enable additional bits for LTSSM state transition tracking while measurement of some events in that category is in progress.

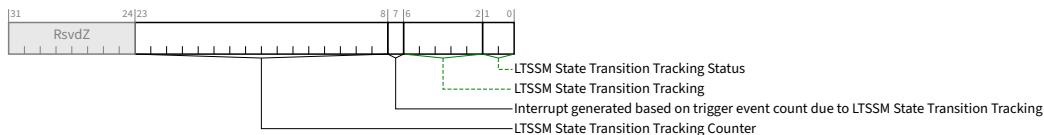


Figure 7-211 LTSSM Performance Measurement Status Register §

Table 7-193 LTSSM Performance Measurement Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 1:0 | LTSSM State Transition Tracking Status <ul style="list-style-type: none"> 00b Not started 01b Started 10b Completed 11b Error (including counter overflow) <p>Cleared on 0b to 1b transition of Flit Latency Measurement Enable . Default is Zero .</p> | ROS |
| 6:2 | LTSSM State Transition Tracking – number of LTSSM state transitions of the measured type tracked so far. | ROS |

201. Earlier versions of this specification had incorrect values for the measurement unit (either 64 µs or 8 µs depending on version). Since expected latency is much lower software can detect such implementations by noticing this value contains Zero.

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>This number does not roll over.</p> <p>Cleared on 0b to 1b transition of Flit Latency Measurement Enable .</p> <p>Default is <u>Zero</u> .</p> | |
| 7 | <p>Interrupt generated based on trigger event count due to LTSSM State Transition Tracking – this bit is Set when an interrupt is generated based on the trigger event. While this bit is Set, no new interrupts will be generated based on the trigger event.</p> <p>Default is <u>Zero</u> .</p> | RW1CS |
| 23:8 | <p>LTSSM State Transition Tracking Counter</p> <p>The measurement unit is 64 usec.</p> <p>Cleared on 0b to 1b transition of Flit Latency Measurement Enable .</p> <p>Default is <u>Zero</u> .</p> | ROS |

7.8.13 Flit Error Injection Extended Capability §

This capability is optional. This capability is permitted in Downstream Ports, in Function 0 of an Upstream Port, and in RCRBs. This capability is not permitted in other Functions.

This capability is only used in Flit Mode. The capability has no effect in Non-Flit Mode.

§ Figure 7-212 details allocation of the register bits in the Flit Error Injection Extended Capability structure.

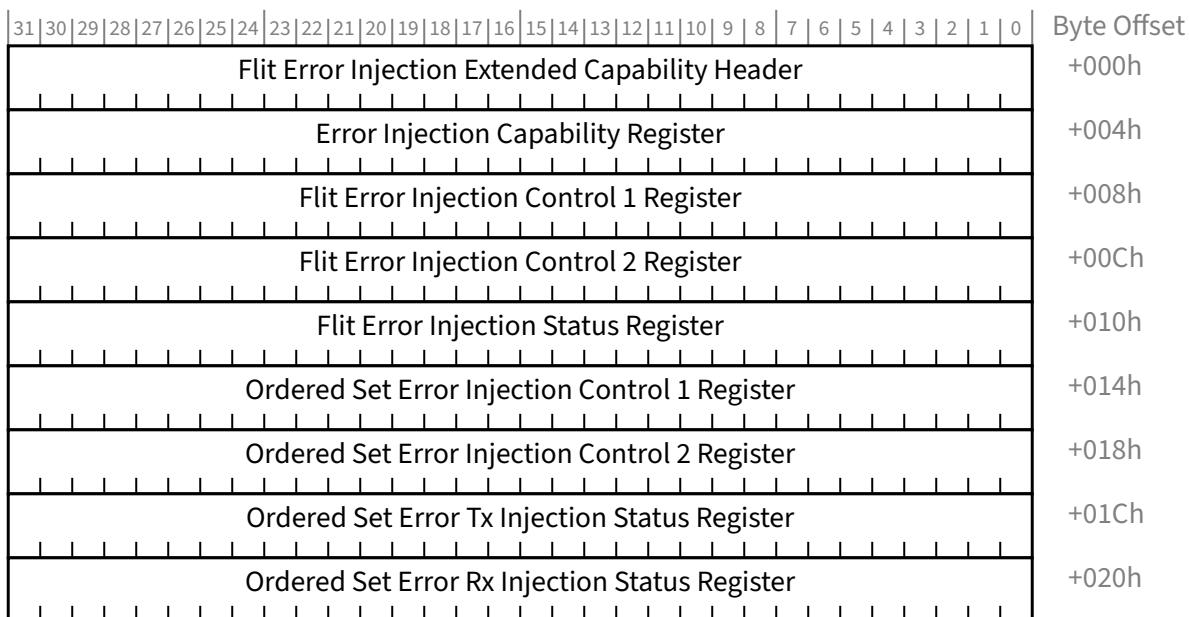
IMPLEMENTATION NOTE: USE CAUTION WHEN INJECTING TOO MANY UNCORRECTABLE ERRORS INTO A FLIT §

↑↑Flit Forward Error Correction (FEC) can detect and correct at most one symbol in error in each of the three ECC groups in a Flit. The Flit CRC is guaranteed to detect up to 8 Symbols in error after FEC correction. When more than 8 Symbols are in error after FEC correction,

Errata: Base 6.3
B843△◀

there is a slight possibility, 2↑↑-64↑↑, ↑↑that the received Flit CRC will match the calculated Flit CRC. This is known as CRC aliasing. When injecting errors into a Flit, one must be aware of the possibility of CRC aliasing. For example, if uncorrectable errors are injected with Error Offset within Flit set to 15 (i.e., Flit byte 0, 15, 30, ..., 225 are corrupted), the FEC decoder will indicate a single error, and the calculated CRC for the Flit will match the received CRC. To avoid such cases when injecting errors, it is recommended to inject no more than 8 Symbol errors into any given Flit.↑

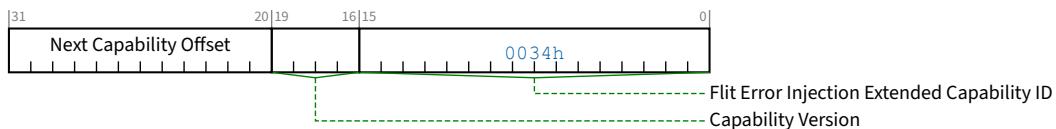
Base 6.4 vs Base 6.3



[Figure 7-212 Flit Error Injection Extended Capability Structure](#)

7.8.13.1 Flit Error Injection Extended Capability Header (Offset 00h)

§ Figure 7-213 details allocation of the register fields in the Flit Error Injection Extended Capability Header ; § Table 7-194 provides the respective bit definitions.



[Figure 7-213 Flit Error Injection Extended Capability Header](#)

[Table 7-194 Flit Error Injection Extended Capability Header](#)

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | Flit Error Injection Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Flit Error Injection Extended Capability is 0034h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:20 | <p>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.</p> <p>For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.</p> | RO |

7.8.13.2 Flit Error Injection Capability Register (Offset 04h) [§](#)



Figure 7-214 Flit Error Injection Capability Register [§](#)

Table 7-195 Flit Error Injection Capability Register [§](#)

| Bit Location | Register Description | Attributes |
|--------------|----------------------|------------|
| 31:0 | Reserved | RsvdP |

7.8.13.3 Flit Error Injection Control 1 Register (Offset 08h) [§](#)

Link level, optional register, both on Tx side as well as Rx side. Behavior is undefined if bits 31:1 of this register change value when error injection is running (Flit Error Injection Enable is 1b and Flit Error Injection Status is 00b or 01b).

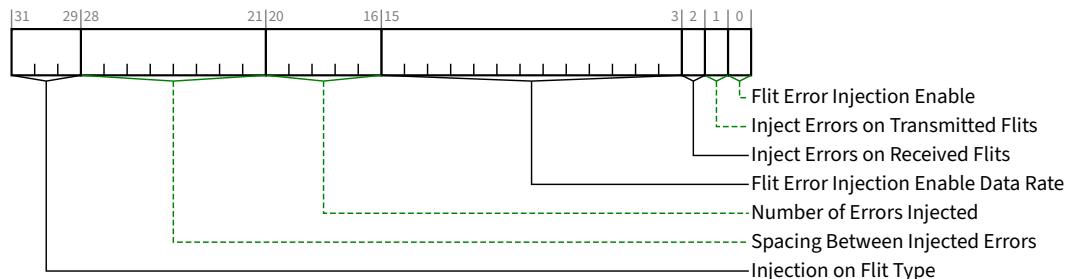


Figure 7-215 Flit Error Injection Control 1 Register [§](#)

Table 7-196 Flit Error Injection Control 1 Register [§](#)

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Flit Error Injection Enable – Setting this bit enables and starts the error injection in the Link. Clearing to this bit stops the error injection.</p> <p>Default Zero .</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1 | Inject Errors on Transmitted Flits – Setting this bit to 1b enables error injection in the Transmitted Flits. A Port that does not implement this functionality must hardwire this bit to 0b. Default is Zero . | RW |
| 2 | Inject Errors on Received Flits – Setting this bit enables error injection in the Received Flits. A Port is permitted to not inject the exact error described but mimic an error injection effect to achieve the desired effect such as logging FEC-correctable errors or causing NAKs after CRC check. A Port that does not implement this functionality must hardwire this bit to 0b. Default is Zero . | RW |
| 15:3 | Flit Error Injection Enable Data Rate – These bits enable the Flit error injection for the corresponding data rates when enabled Bit 3 2.5 GT/s Bit 4 5.0 GT/s Bit 5 8.0 GT/s Bit 6 16.0 GT/s Bit 7 32.0 GT/s Bit 8 64.0 GT/s Bits 15:9 RsvdP Default is Zero . | RW / RsvdP |
| 20:16 | Number of Errors Injected – This represents the number of errors to be injected on the Transmitted and/or Received Flits independently. A value of 0 indicates that error injection continues till the injection mechanism is disabled. Default is Zero . | RW |
| 28:21 | Spacing Between Injected Errors – This represents the next Flit on which error will be injected after the current sequence of Flit error injection completes. A non-0 value indicates the exact number of Flits after which the error is injected; a 0 value will inject the errors after a pseudo-random number of Flits between 1 to 127, chosen with equal probability. This is used on the Transmit and/or Received side independently. Default is Zero . | RW |
| 31:29 | Injection on Flit Type – 000b Inject on any Flit Type 001b Inject on any non-IDLE Flit 010b Inject only on Payload Flit 011b Inject only on NOP Flit 100b Inject only on IDLE Flit 101b If Error Type Being Injected is 11b, Inject only on a Payload Flit and then subsequently on the same sequence number for the Consecutive Error Injection times. The entire repeat will be considered as one instance of error injection for the purposes of counting towards the number of errors injected. Reserved encoding if Error Type Being Injected is other than 11b. 110b If Error Type Being Injected is 11b, Inject only on a Payload Flit along with subsequent injection on one selected Payload Flit with the same sequence number for Consecutive Error Injection times. Exactly one sequence number is selected for consecutive error injection among all the outstanding Payload Flits injected with FEC-uncorrectable errors. The entire repeat will be considered as one instance of error injection for the purposes of counting | RW |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>towards the number of errors injected. Reserved encoding if <u>Error Type Being Injected</u> is other than 11b.</p> <p>111b Reserved</p> <p>Default is Zero</p> | |

7.8.13.4 Flit Error Injection Control 2 Register (Offset 0Ch) §

Link level, optional register, both on Tx side as well as Rx side. Behavior is undefined if this register changes value when error injection is running (Flit Error Injection Enable is 1b and Flit Error Injection Status is 00b or 01b).

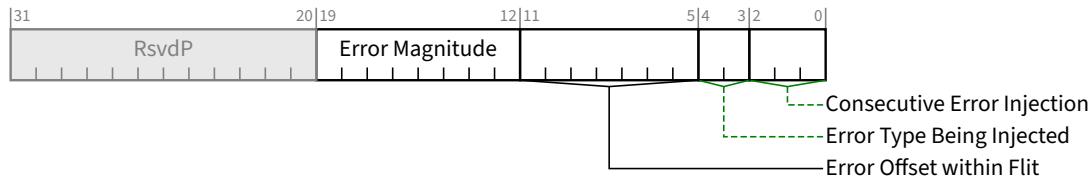


Figure 7-216 Flit Error Injection Control 2 Register §

Table 7-197 Flit Error Injection Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 2:0 | <p>Consecutive Error Injection – The number of consecutive Flits that will be injected with the error, irrespective of the type of Flit on which the error is supposed to be initially injected. For the Injection of Flit Type encoding of 101b and 110b, this field has additional meaning, as described above. Even if multiple consecutive Flits will be injected with an error because of this, the entire sequence will count as one towards the number of errors injected.</p> <p>000b No consecutive error injection</p> <p>001b to 110b One to six consecutive error injections</p> <p>111b A pseudo-random number between 7 and 15, each selected with equal probability</p> <p>Default is Zero .</p> | RW |
| 4:3 | <p>Error Type Being Injected –</p> <p>00b Random between FEC-correctable or FEC-uncorrectable</p> <p>01b FEC-Correctable error injected only in one FEC group (rotate across the groups in subsequent injections)</p> <p>10b FEC-Correctable error injected in all 3 FEC groups simultaneously</p> <p>11b FEC-Uncorrectable error injected</p> <p>Default is Zero .</p> | RW |
| 11:5 | <p>Error Offset within Flit – For FEC-correctable error(s): Byte offset within FEC-group where error will be injected. If this value is greater than the number of bytes in the FEC group, an error must be injected on any byte in the FEC-group with equal probability using a pseudo-random number generator.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | For uncorrectable error(s): Distance between subsequent injected error bytes with the initial starting position at byte 0. If at least 8 bytes have not been injected with an error, the Port must inject errors in some of the FEC bytes to get to 8 bytes in error. Default is Zero . | |
| 19:12 | Error Magnitude – magnitude of error injected in each byte where error has been injected 00h any pseudo-random non-0 value with equal probability Others exact error magnitude Default is Zero . | RW |

7.8.13.5 Flit Error Injection Status Register (Offset 10h) §

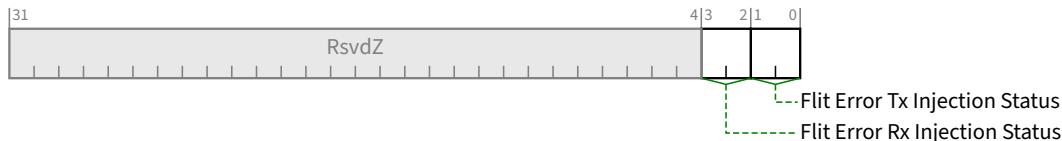


Figure 7-217 Flit Error Injection Status Register §

Table 7-198 Flit Error Injection Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 1:0 | Flit Error Tx Injection Status 00b Not injected any error yet 01b At least one error is injected, but not completed 10b Error injection completed 11b Error case – error injection aborted, either Flit Error Injection Enable was cleared while injection was incomplete or optionally Flit Error Injection Control 1 [31:1] or Flit Error Injection Control 2 was changed while Injection was enabled and not complete. This field is cleared on the 0b to 1b transition of Flit Error Injection Enable . Default is Zero . | RO |
| 3:2 | Flit Error Rx Injection Status 00b Not injected any error yet 01b At least one error is injected, but not completed 10b Error injection completed 11b Error case – error injection aborted, either Flit Error Injection Enable was cleared while injection was incomplete or optionally Flit Error Injection Control 1 [31:1] or Flit Error Injection Control 2 was changed while Injection was enabled and not complete. This field is cleared on the 0b to 1b transition of Flit Error Injection Enable . Default is Zero . | RO |

7.8.13.6 Ordered Set Error Injection Control 1 Register (Offset 14h) §

Link level, optional register, both on Tx side as well as Rx side. A Port that does not implement the functionality must hardwire these bits to 0b. Behavior is undefined if bits 63:1 of this register change value when Ordered Set Injection Enable is Set and Ordered Set Error Injection Status is 00b or 01b.

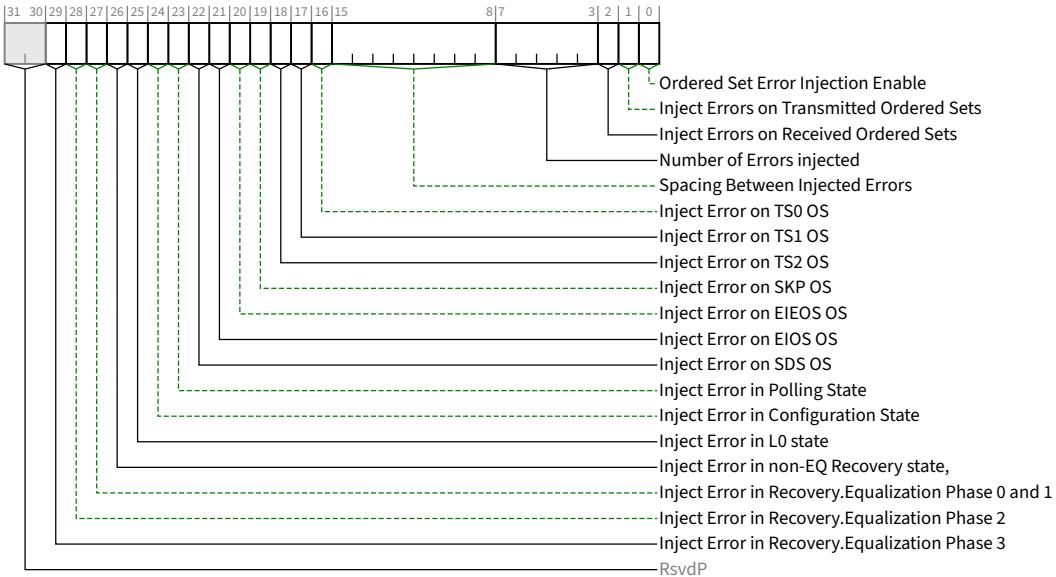


Figure 7-218 Ordered Set Error Injection Control 1 Register §

Table 7-199 Ordered Set Error Injection Control 1 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Ordered Set Error Injection Enable – Setting this bit enables and starts Error Injection on the Link. Clearing this bit stops the error injection.</p> <p>Behavior is undefined if <u>Ordered Set Error Injection Enable</u> is Set and <u>Inject Errors on Transmitted Ordered Sets</u> and <u>Inject Errors on Received Ordered Sets</u> are both Clear.</p> <p>Default is Zero .</p> | RWS |
| 1 | <p>Inject Errors on Transmitted Ordered Sets – Setting this bit to 1b enables error injection in the Transmitted Ordered Sets. A Port that does not implement this functionality must hardwire this bit to 0b.</p> <p>Behavior is undefined if <u>Ordered Set Error Injection Enable</u> , <u>Inject Errors on Transmitted Ordered Sets</u> , and <u>Inject Errors on Received Ordered Sets</u> are all Set.</p> <p>Default is Zero .</p> | RWS |
| 2 | <p>Inject Errors on Received Ordered Sets – Setting this bit enables error injection in the Received Ordered Sets. A Port is permitted to not inject the exact error described but treat the Ordered Set as invalid. A Port that does not implement this functionality must hardwire this bit to 0b.</p> <p>Behavior is undefined if <u>Ordered Set Error Injection Enable</u> , <u>Inject Errors on Transmitted Ordered Sets</u> , and <u>Inject Errors on Received Ordered Sets</u> are all Set.</p> <p>Default is Zero .</p> | RWS |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:3 | Number of Errors injected – This represents the number of errors to be injected. A value of 0 indicates that error injection continues till the injection mechanism is disabled. Default is Zero . | RWS |
| 15:8 | Spacing Between Injected Errors – This represents the next OS on which error will be injected after the current OS error injection completes. A non-0 value indicates the exact number of OSs after which the error is injected; a 0 value will inject the errors after a pseudo-random number of OSs between 1 to 127, chosen with equal probability. This is used on the Transmit and/or Received side independently. Default is Zero . | RWS |
| 16 | Inject Error on TSO OS – When Set, injects errors on <u>TSO</u> OS. | RWS |
| 17 | Inject Error on TS1 OS – When Set, injects errors on <u>TS1</u> OS. | RWS |
| 18 | Inject Error on TS2 OS – When Set, injects errors on <u>TS2</u> OS. | RWS |
| 19 | Inject Error on SKP OS – When Set, injects errors on <u>SKP</u> OS . | RWS |
| 20 | Inject Error on EIEOS OS – When Set, injects errors on <u>EIEOS</u> OS. | RWS |
| 21 | Inject Error on EIOS OS – When Set, injects errors on <u>EIOS</u> OS. | RWS |
| 22 | Inject Error on SDS OS – When Set, injects errors on <u>SDS</u> OS. | RWS |
| 23 | Inject Error in Polling State – When Set, injects errors in the <u>Polling</u> LTSSM state. | RWS |
| 24 | Inject Error in Configuration State – When Set, injects errors in the <u>Configuration</u> LTSSM state. | RWS |
| 25 | Inject Error in L0 state – When Set, injects errors in the <u>L0</u> LTSSM state. | RWS |
| 26 | Inject Error in non-EQ Recovery state , – When Set, injects errors in the <u>Recovery</u> LTSSM states except for the <u>Recovery.Equalization</u> substate. | RWS |
| 27 | Inject Error in Recovery.Equalization Phase 0 and 1 – When Set, injects errors <u>Recovery.Equalization</u> Phase 0 and Phase 1. | RWS |
| 28 | Inject Error in Recovery.Equalization Phase 2 – When Set, injects errors <u>Recovery.Equalization</u> Phase 2. | RWS |
| 29 | Inject Error in Recovery.Equalization Phase 3 – When Set, injects errors <u>Recovery.Equalization</u> Phase 3. | RWS |

7.8.13.7 Ordered Set Error Injection Control 2 Register (Offset 18h) §

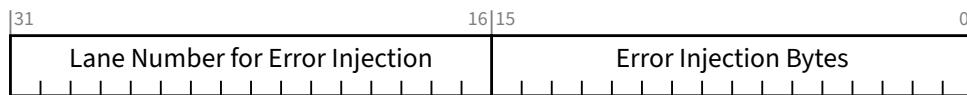


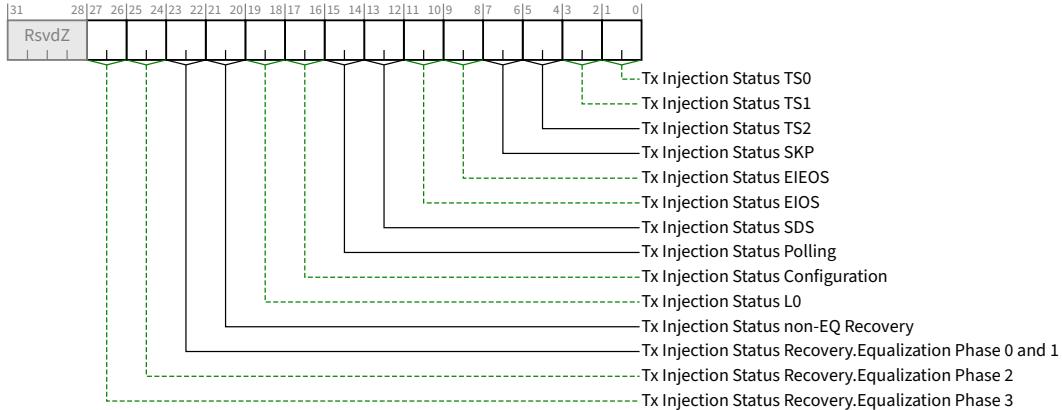
Figure 7-219 Ordered Set Error Injection Control 2 Register §

Table 7-200 Ordered Set Error Injection Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Error Injection Bytes – Individual bytes where errors (any magnitude) will be injected; all 0s indicates a byte will be chosen based on a pseudo-random generator between 1 and 16. For SKP OS, each bit covers 2.5 Bytes instead of one byte | RWS |
| 31:16 | Lane Number for Error Injection – A value of 1b in one or more bit positions indicates that the corresponding Lane number will participate in error injection when enabled. Bit 0 of this field corresponds to Lane 0. | RWS |

7.8.13.8 Ordered Set Error Tx Injection Status Register (Offset 1Ch) §

This register contains a set of fields for Tx Ordered Set Error Injection. The description for Tx Injection Status TS0 applies to all fields in this register.

*Figure 7-220 Ordered Set Tx Error Injection Status Register §**Table 7-201 Ordered Set Tx Error Injection Status Register §*

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 1:0 | Tx Injection Status TS0 – Each two bit field is encoded as follows: <ul style="list-style-type: none"> 00b Not injected any error yet 01b At least one error is injected, but not completed 10b Error injection completed 11b Error case – error injection aborted, either <u>Ordered Set Error Injection Enable</u> was cleared while injection was incomplete or optionally any bit in <u>Ordered Set Error Injection Control 1 [31:1]</u> or <u>Ordered Set Injection Control 2</u> was changed while Injection was enabled and not complete. <p>This field is cleared on the 0b to 1b transition of <u>Ordered Set Error Injection Enable</u>. Default is 00b.</p> | ROS |
| 3:2 | Tx Injection Status TS1 | ROS |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 5:4 | Tx Injection Status TS2 | <u>ROS</u> |
| 7:6 | Tx Injection Status SKP | <u>ROS</u> |
| 9:8 | Tx Injection Status EIEOS | <u>ROS</u> |
| 11:10 | Tx Injection Status EIOS | <u>ROS</u> |
| 13:12 | Tx Injection Status SDS | <u>ROS</u> |
| 15:14 | Tx Injection Status Polling | <u>ROS</u> |
| 17:16 | Tx Injection Status Configuration | <u>ROS</u> |
| 19:18 | Tx Injection Status L0 | <u>ROS</u> |
| 21:20 | Tx Injection Status non-EQ Recovery | <u>ROS</u> |
| 23:22 | Tx Injection Status Recovery.Equalization Phase 0 and 1 | <u>ROS</u> |
| 25:24 | Tx Injection Status Recovery.Equalization Phase 2 | <u>ROS</u> |
| 27:26 | Tx Injection Status Recovery.Equalization Phase 3 | <u>ROS</u> |

7.8.13.9 Ordered Set Error Rx Injection Status Register (Offset 20h) §

This register contains a set of fields for Rx Ordered Set Error Injection. The description for Rx Injection Status TS0 applies to all fields in this register.

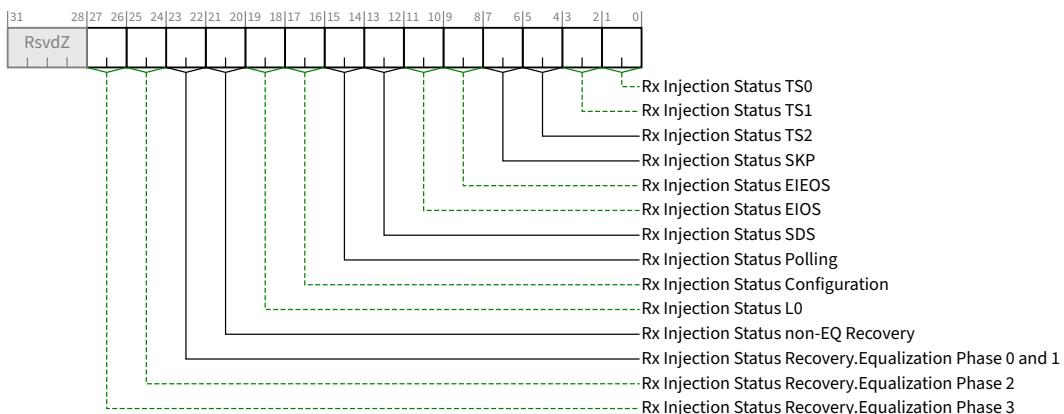


Figure 7-221 Ordered Set Rx Error Injection Status Register §

Table 7-202 Ordered Set Rx Error Injection Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 1:0 | Rx Injection Status TS0 – Each two bit field is encoded as follows: 00b Not injected any error yet | <u>ROS</u> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>01b At least one error is injected, but not completed</p> <p>10b Error injection completed</p> <p>11b Error case – error injection aborted, either <u>Ordered Set Error Injection Enable</u> was cleared while injection was incomplete or optionally any bit in Ordered Set Error Injection Control 1 [31:1] or Ordered Set Injection Control 2 was changed while Injection was enabled and not complete.</p> <p>This field is cleared on the 0b to 1b transition of <u>Ordered Set Error Injection Enable</u>.</p> <p>Default is 00b.</p> | |
| 3:2 | Rx Injection Status TS1 | ROS |
| 5:4 | Rx Injection Status TS2 | ROS |
| 7:6 | Rx Injection Status SKP | ROS |
| 9:8 | Rx Injection Status EIEOS | ROS |
| 11:10 | Rx Injection Status EIOS | ROS |
| 13:12 | Rx Injection Status SDS | ROS |
| 15:14 | Rx Injection Status Polling | ROS |
| 17:16 | Rx Injection Status Configuration | ROS |
| 19:18 | Rx Injection Status L0 | ROS |
| 21:20 | Rx Injection Status non-EQ Recovery | ROS |
| 23:22 | Rx Injection Status Recovery.Equalization Phase 0 and 1 | ROS |
| 25:24 | Rx Injection Status Recovery.Equalization Phase 2 | ROS |
| 27:26 | Rx Injection Status Recovery.Equalization Phase 3 | ROS |

7.8.14 NOP Flit Extended Capability §

The NOP Flit Extended Capability is an optional Extended Capability that provides control over NOP Flit usage by a transmitting port.

This capability is permitted in Downstream Ports, in Function 0 of an Upstream Port, and in RCRBs. This capability is not permitted in other Functions.

VFs must not implement this capability.

The NOP Flit Extended Capability structure is shown in § [Figure 7-222](#).

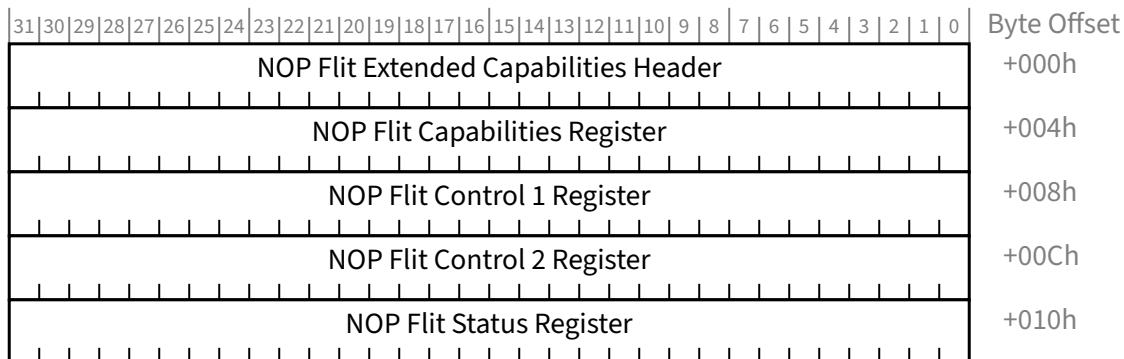


Figure 7-222 NOP Flit Extended Capability §

7.8.14.1 NOP Flit Extended Capability Header §

§ Figure 7-223 details allocation of register fields in the NOP Flit Extended Capability Header; § Table 7-203 provides the respective bit definitions.

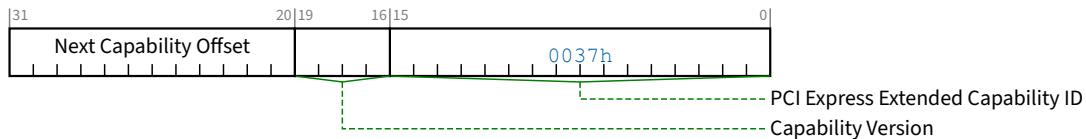


Figure 7-223 NOP Flit Extended Capability Header §

Table 7-203 NOP Flit Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the NOP Flit Extended Capability is 0037h . | HwInit |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h. | HwInit |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | HwInit |

7.8.14.2 NOP Flit Capabilities Register §

§ Figure 7-224 details allocation of register fields in the NOP Flit Capabilities Register ; § Table 7-204 provides the respective bit definitions.

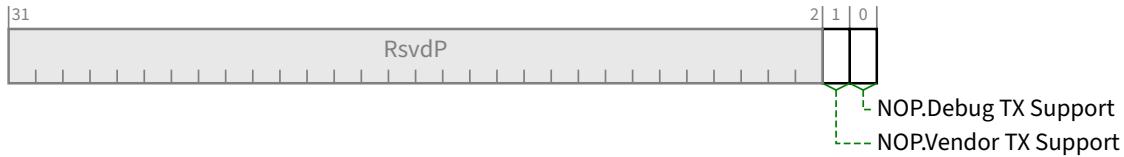


Figure 7-224 NOP Flit EditorialCapabilites EditorialCapabilities Register §

Table 7-204 NOP Flit ~~Capabilities~~ EditorialCapabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| 0 | NOP.Debug TX Support – Indicates transmitter support for sending NOP.Debug Flits | <u>HwInit</u> |
| 1 | NOP.Vendor TX Support – Indicates transmitter support for sending NOP.Vendor Flits | <u>HwInit</u> |

7.8.14.3 NOP Flit Control 1 Register §

§ Figure 7-225 details allocation of register fields in the NOP Flit Control 1 Register ; § Table 7-205 provides the respective bit definitions.

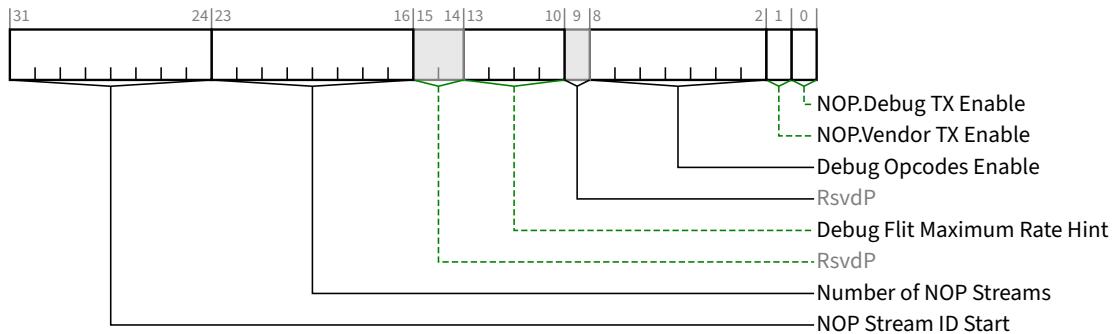


Figure 7-225 NOP Flit Control 1 Register §

Table 7-205 NOP Flit Control 1 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-----------------------|
| 0 | NOP.Debug TX Enable – When Set, this bit enables capable transmitters of sending NOP.Debug Flits. When Clear, the transmitter must not send any NOP.Debug Flits. | RWS / <u>RsvdP</u> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| | <p>This bit is RsvdP when the <u>NOP.Debug TX Support</u> bit is Clear.</p> <p>Default value of this bit is implementation specific.</p> | |
| 1 | <p>NOP.Vendor TX Enable – When Set, this bit enables capable transmitters of sending <u>NOP.Vendor Flits</u>. When Clear, the transmitter must not send any <u>NOP.Vendor Flits</u>.</p> <p>This bit is RsvdP when the NOP.Vendor TX Support bit is Clear.</p> <p>Default value of this bit is implementation specific.</p> | RWS / RsvdP |
| 8:2 | <p>Debug Opcodes Enable – When the NOP.Debug TX Enable bit is Set, this field controls which PCI-SIG defined Debug Opcodes may be sent by the transmitter. If this field is Zero, the transmitter may send any PCI-SIG defined Debug Opcode; otherwise, the transmitter may only send the PCI-SIG defined Debug Opcode encoding programmed into this field. Control over non-PCI-SIG defined Debug Opcodes is vendor specific.</p> <p>This bit is RsvdP when the <u>NOP.Debug TX Support</u> bit is Clear.</p> <p>Default value of this field is Zero.</p> | RWS / RsvdP |
| 13:10 | <p>Debug Flit Maximum Rate Hint – Controls the desired maximum rate of NOP.Debug Flit injection by the transmitter for periodic Debug Opcodes not triggered by hardware events. Non-periodic NOP.Debug Flits that are triggered by hardware events are not limited by this setting and may be scheduled independently. A transmitter is permitted to inject at a lower or higher rate than the maximum rate in this field.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0h Link rate (one every Flit) 1h Link rate / 2 (one every two Flits) 2h Link rate / 3 (one every three Flits) 3h Link rate / 4 (one every four Flits) 4h Link rate / 5 (one every five Flits) 5h Link rate / 6 (one every six Flits) 6h Link rate / 7 (one every seven Flits) 7h Link rate / 8 (one every eight Flits) 8h Link rate / 9 (one every nine Flits) 9h Link rate / 10 (one every ten Flits) Others All other encodings are Reserved. <p>This field is RsvdP when the <u>NOP.Debug TX Support</u> bit is Clear.</p> <p>Default value of this field is 3h.</p> | RWS / RsvdP |
| 23:16 | <p>Number of NOP Streams – When either or both the <u>NOP.Debug TX Enable</u> bit or the <u>NOP.Vendor TX Enable</u> bit are Set, this field controls the number of NOP Stream IDs the Transmitter may use for sending NOP Flits. The field's value is the total number of NOP Streams minus one. A value of 0 indicates support for one NOP Stream.</p> <p>The upper end NOP Stream ID value is defined as <u>NOP Stream ID Start</u> + <u>Number of NOP Streams</u>, with an upper limit of FFh. For example, if the <u>Number of NOP Streams</u> field contains 5h and the <u>NOP Stream ID Start</u> field contains FEh, the range of Transmitter usable NOP Stream IDs would only be FEh and FFh (i.e., two NOP Stream IDs), even though the <u>Number of NOP Streams</u> field value was programmed to be greater than that.</p> <p>This field is only used when originating NOP Flits and has no effect on forwarding NOP Flits between Ports.</p> <p>This field is RsvdP when the <u>NOP.Vendor TX Support</u> and the <u>NOP.Debug TX Support</u> bits are both Clear.</p> | RWS / RsvdP |

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| | Default value of this field is 0. | |
| 31:24 | <p>NOP Stream ID Start – When either or both the NOP.Debug TX Enable bit or the NOP.Vendor TX Enable bit are Set, this field controls the lower end of the range of NOP Stream IDs that the Transmitter may use for sending NOP Flits.</p> <p>This field is only used when originating NOP Flits and has no effect on forwarding NOP Flits between Ports.</p> <p>This field is RsvdP when the NOP.Vendor TX Support and the NOP.Debug TX Support bits are both Clear.</p> <p>Default value of this field is FFh.</p> | RWS / RsvdP |

7.8.14.4 NOP Flit Control 2 Register §

§ Figure 7-226 details allocation of register fields in the NOP Flit Control 2 Register ; § Table 7-206 provides the respective bit definitions.

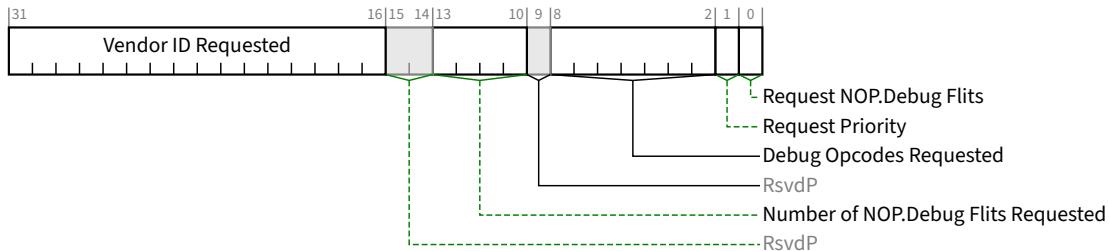


Figure 7-226 NOP Flit Control 2 Register §

Table 7-206 NOP Flit Control 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------------------|
| 0 | <p>Request NOP.Debug Flits – When NOP.Debug TX Enable is Set, this bit requests NOP.Debug Flits to be initiated by a transmitter. A write of 1b to this bit initiates the request so that the transmitter samples the Request Priority, the Debug Opcode Requested , the Number of NOP.Debug Flits Requested , and the Vendor ID Requested fields and attempts to fulfill the request on a best effort basis.</p> <p>It is permitted to write 1b to this bit while simultaneously writing modified values to other fields in this register. The resulting request must use the modified values.</p> <p>Hardware behavior is undefined if this is written while the NOP.Debug Flit Request in Progress bit is Set.</p> <p>This bit will always return 0b when read.</p> <p>This bit is RsvdP when NOP.Debug TX Support is Clear.</p> <p>Default value of this bit is Zero.</p> | RWS / RsvdP (see description) |
| 1 | <p>Request Priority – When the NOP.Debug TX Enable bit is Set, this bit controls the priority of the requested NOP.Debug Flits. If this bit is Set, the transmitter must prioritize the requested NOP.Debug Flits over any periodic NOP.Debug Flit transmissions. If this bit is Clear, the transmitter need not prioritize the requested NOP.Debug Flits over any periodic NOP.Debug Flit transmissions.</p> <p>This bit is RsvdP when NOP.Debug TX Support is Clear.</p> | RWS / RsvdP |

| Bit Location | Register Description | Attributes |
|--------------|---|-------------|
| | Default value of this bit is Zero. | |
| 8:2 | <p>Debug Opcodes Requested – When the NOP.Debug TX Enable bit is Set, this field controls the number of NOP.Debug Flits which Debug Opcode is requested for transmission. Errata: Base 6.3 B826△</p> <p>This bit is RsvdP when the NOP.Debug TX Support bit is Clear.</p> <p>Default value of this field is Zero.</p> | RWS / RsvdP |
| 13:10 | <p>Number of NOP.Debug Flits Requested – When the NOP.Debug TX Enable bit is Set, this field controls the number of NOP.Debug Flits requested for transmission.</p> <p>This bit is RsvdP when the NOP.Debug TX Support bit is Clear.</p> <p>Default value of this field is Zero.</p> | RWS / RsvdP |
| 31:16 | <p>Vendor ID Requested – When the NOP.Debug TX Enable bit is Set, this field controls which Vendor ID associated with the Debug Opcode is requested for transmission.</p> <p>This field is RsvdP when the NOP.Debug TX Support bit is Clear.</p> <p>Default value of this field is Zero.</p> | RWS / RsvdP |

7.8.14.5 NOP Flit Status Register §

§ Figure 7-227 details allocation of register fields in the NOP Flit Status Register ; § Table 7-207 provides the respective bit definitions.

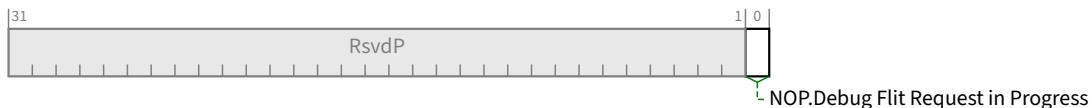


Figure 7-227 NOP Flit Status Register §

Table 7-207 NOP Flit Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>NOP.Debug Flit Request in Progress – When Set, this bit indicates that the Transmitter has started to fulfill the requested NOP.Debug Flits and that the request has not yet been fully completed. A Transmitter reports this bit Clear only when the request has been fully completed or the request has been cancelled. This bit must also be Cleared when the NOP.Debug TX Enable bit is Cleared.</p> <p>Ports that do not implement the ability to transmit the requested NOP.Debug Flits are permitted to hardwire this bit to 0b.</p> <p>This bit is RsvdZ when the NOP.Debug TX Support bit is Clear.</p> <p>Default value of this bit is 0.</p> | RO / RsvdZ |

7.9 Additional PCI and PCIe Capabilities §

This section, contains a description of additional PCI and PCIe capabilities that are individually optional in this but may be required by other PCISIG specifications.

7.9.1 Virtual Channel Extended Capability §

The Virtual Channel Extended Capability (**VC Extended Capability**) is an optional Extended Capability required for devices that have Ports (or for individual Functions) that support functionality beyond the default Traffic Class (TC0) over the default Virtual Channel (VC0). This may apply to devices with only one VC that support TC filtering or to devices that support multiple VCs. Note that a PCI Express device that supports only TC0 over VC0 does not require VC Extended Capability and associated registers. § Figure 7-228 provides a high level view of the Virtual Channel Extended Capability structure. This structure controls Virtual Channel assignment for PCI Express Links and may be present in any device (or RCRB) that contains (controls) a Port, or any device that has a Multi-Function Virtual Channel (MFVC) Capability structure. Some registers/fields in the Virtual Channel Extended Capability structure may have different interpretation for Endpoints, Switch Ports, Root Ports and RCRB . Software must interpret the Device/Port Type field in the PCI Express Capabilities register to determine the availability and meaning of these registers/fields.

The number of (extended) Virtual Channels is indicated by the Extended VC Count field in the Port VC Capability Register 1 . Software must interpret this field to determine the availability of extended VC Resource registers.

The VC Extended Capability structure is permitted in the Extended Configuration Space of all Single-Function Devices or in RCRBs .

Each **SR-IOV** VF **or SIOV SDI** uses the Virtual Channel **capabilities** of its associated PF. **VFs themselves must not contain any Virtual Channel Capabilities.** **permitted to have a Virtual Channel Extended Capability structure in their Extended Configuration Space.**

ECN: Base 6.3
SIOV△
—

A Multi-Function Device at an Upstream Port is permitted to contain a Multi-Function Virtual Channel (MFVC) Capability structure (see § Section 7.9.2). If a Multi-Function Device contains an MFVC Capability structure, any or all of its Functions with the exception of VFs are permitted to contain a VC Extended Capability structure. Per-Function VC Extended Capability structures are also permitted for devices inside a Switch that contain only Switch Downstream Port Functions, or for RCiEPs. Otherwise, only Function 0 is permitted to contain a VC Extended Capability structure.

To preserve software backward compatibility, two Extended Capability IDs are permitted for VC Extended Capability structures: 0002h and 0009h. Any VC Extended Capability structure in a device that also contains an MFVC Capability structure must use the Extended Capability ID 0009h . A VC Extended Capability structure in a device that does not contain an MFVC Capability structure must use the Extended Capability ID 0002h .

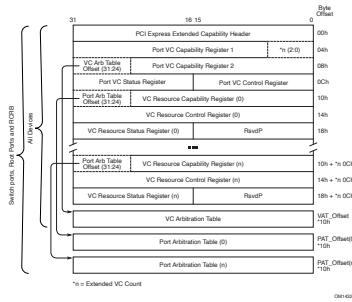


Figure 7-228 Virtual Channel Extended Capability Structure §

The following sections describe the registers/fields of the Virtual Channel Extended Capability structure.

7.9.1.1 Virtual Channel Extended Capability Header (Offset 00h) §

Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability header. A Virtual Channel Extended Capability must use one of two Extended Capability IDs: 0002h or 0009h. Refer to § Section 7.9.1 for rules governing when each should be used. § Figure 7-229 details allocation of register fields in the Virtual Channel Extended Capability Header ; § Table 7-208 provides the respective bit definitions.

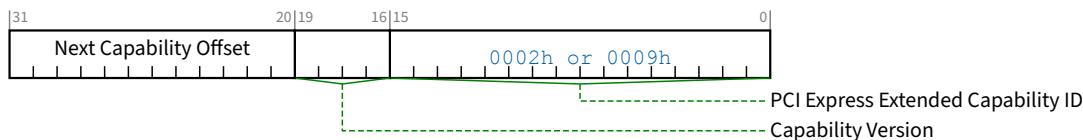


Figure 7-229 Virtual Channel Extended Capability Header §

Table 7-208 Virtual Channel Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Virtual Channel Extended Capability is either 0002h or 0009h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.9.1.2 Port VC Capability Register 1 (Offset 04h) §

The Port VC Capability Register 1 describes the configuration of the Virtual Channels associated with a PCI Express Port. § Figure 7-230 details allocation of register fields in the Port VC Capability Register 1 ; § Table 7-209 provides the respective bit definitions.

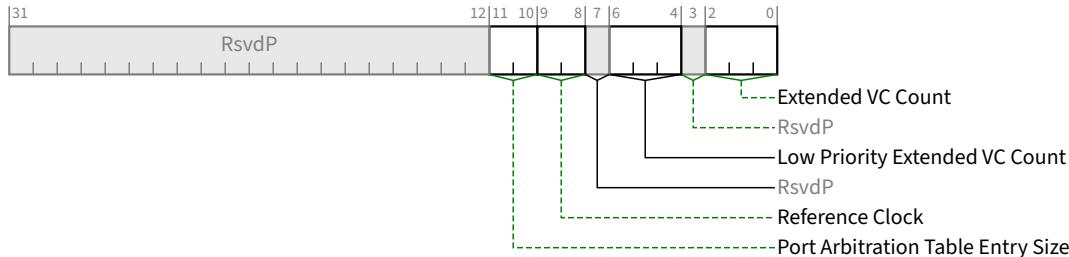


Figure 7-230 Port VC Capability Register 1 §

Table 7-209 Port VC Capability Register 1 §

| Bit Location | Register Description | Attributes | | | | |
|------------------|--|------------|--|------------------|---|----|
| 2:0 | <p>Extended VC Count - Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device. This field is valid for all Functions.</p> <p>This value indicates the number of (extended) VC Resource Capability, Control, and Status registers that are present in Configuration Space in addition to the required VC Resource registers for the default VC.</p> <p>The minimum value of this field is 0 (for devices that only support the default VC and only have 1 set of VC Resource Registers for that VC). The maximum value is 7.</p> | RO | | | | |
| 6:4 | <p>Low Priority Extended VC Count - Indicates the number of (extended) Virtual Channels in addition to the default VC belonging to the low-priority VC (LPVC) group that has the lowest priority with respect to other VC resources in a strict-priority VC Arbitration. This field is valid for all Functions.</p> <p>The minimum value of this field is 000b and the maximum value is Extended VC Count .</p> | RO | | | | |
| 9:8 | <p>Reference Clock - Indicates the reference clock for Virtual Channels that support time-based WRR Port Arbitration. This field is valid for RCRBs , Switch Ports, and Root Ports that support peer-to-peer traffic. It is not valid for Root Ports that do not support peer-to-peer traffic, Endpoints, and Switches or Root Complexes not implementing WRR, and must be hardwired to 00b.</p> <p>Defined encodings are:</p> <table> <tr> <td>00b</td> <td>100 ns reference clock</td> </tr> <tr> <td>01b - 11b</td> <td>Reserved</td> </tr> </table> | 00b | 100 ns reference clock | 01b - 11b | Reserved | RO |
| 00b | 100 ns reference clock | | | | | |
| 01b - 11b | Reserved | | | | | |
| 11:10 | <p>Port Arbitration Table Entry Size - Indicates the size (in bits) of Port Arbitration table entry in the Function. This field is valid only for RCRBs , Switch Ports, and Root Ports that support peer-to-peer traffic. It is not valid and must be hardwired to 00b for Root Ports that do not support peer-to-peer traffic and Endpoints.</p> <p>Defined encodings are:</p> <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> </table> | 00b | The size of Port Arbitration table entry is 1 bit. | 01b | The size of Port Arbitration table entry is 2 bits. | RO |
| 00b | The size of Port Arbitration table entry is 1 bit. | | | | | |
| 01b | The size of Port Arbitration table entry is 2 bits. | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | 10b The size of Port Arbitration table entry is 4 bits. | |
| | 11b The size of Port Arbitration table entry is 8 bits. | |

7.9.1.3 Port VC Capability Register 2 (Offset 08h) §

The Port VC Capability Register 2 provides further information about the configuration of the Virtual Channels associated with a PCI Express Port. § Figure 7-231 details allocation of register fields in the Port VC Capability Register 2 ; § Table 7-210 provides the respective bit definitions.

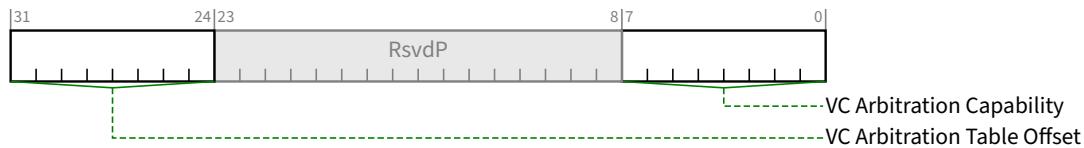


Figure 7-231 Port VC Capability Register 2 §

Table 7-210 Port VC Capability Register 2 §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | <p>VC Arbitration Capability - Indicates the types of VC Arbitration supported by the Function for the LPVC group. This field is valid for all Functions that report a Low Priority Extended VC Count field greater than 0. For all other Functions, this field must be hardwired to 00h.</p> <p>Each Bit Location within this field corresponds to a VC Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the Port can be configured to provide different VC arbitration services.</p> <p>Defined bit positions are:</p> <ul style="list-style-type: none"> Bit 0 Hardware fixed arbitration scheme, e.g., Round Robin Bit 1 Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 WRR arbitration with 64 phases Bit 3 WRR arbitration with 128 phases Bits 4-7 Reserved | RO |
| 31:24 | <p>VC Arbitration Table Offset - Indicates the location of the VC Arbitration Table. This field is valid for all Functions.</p> <p>This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Extended Capability structure. A value of 0 indicates that the table is not present.</p> | RO |

7.9.1.4 Port VC Control Register (Offset 0Ch) §

§ Figure 7-232 details allocation of register fields in the Port VC Control Register ; § Table 7-211 provides the respective bit definitions.

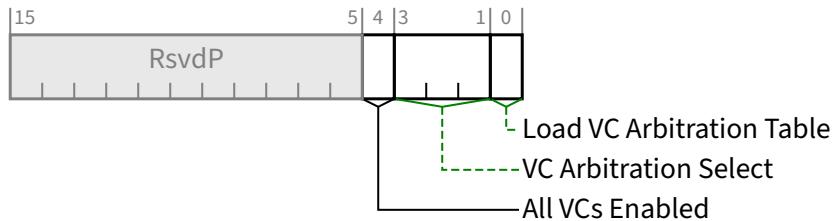


Figure 7-232 Port VC Control Register §

Table 7-211 Port VC Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Load VC Arbitration Table - Used by software to update the VC Arbitration Table. This bit is valid for all Functions when the selected VC Arbitration uses the VC Arbitration Table.</p> <p>Software sets this bit to request hardware to apply new values programmed into VC Arbitration Table; clearing this bit has no effect. Software checks the VC Arbitration Table Status bit to confirm that new values stored in the VC Arbitration Table are latched by the VC arbitration logic.</p> <p>This bit always returns 0b when read.</p> | RW |
| 3:1 | <p>VC Arbitration Select - Used by software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration Capability field in the <u>Port VC Capability Register 2</u>. This field is valid for all Functions.</p> <p>The permissible values of this field are numbers corresponding to one of the asserted bits in the VC Arbitration Capability field.</p> <p>This field cannot be modified when more than one VC in the LPVC group is enabled.</p> | RW |
| 4 | <p>All VCs Enabled - Setting this bit indicates that all VCs that will be used by the Port have been enabled. Setting this bit allows hardware to allocate assigned buffer resources across the enabled VCs.</p> <p>Setting this bit is optional. If this bit remains Clear and some VC Resources are never enabled, performance may be affected but the Link and all enabled VCs must operate correctly.</p> <p>Behavior is undefined if this bit is Set and any VC Enable bit in this capability changes value.</p> <p>Default value of this bit is 0b.</p> | RW |

7.9.1.5 Port VC Status Register (Offset 0Eh) §

The Port VC Status Register provides status of the configuration of Virtual Channels associated with a Port. § Figure 7-233 details allocation of register fields in the Port VC Status Register; § Table 7-212 provides the respective bit definitions.

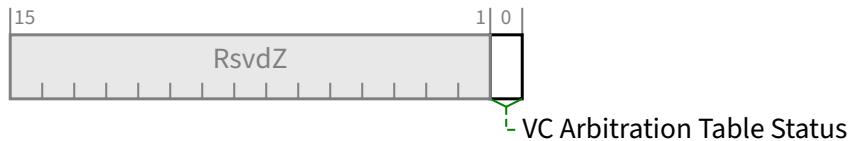


Figure 7-233 Port VC Status Register §

Table 7-212 Port VC Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>VC Arbitration Table Status - Indicates the coherency status of the VC Arbitration Table. This bit is valid for all Functions when the selected VC uses the VC Arbitration Table.</p> <p>This bit is Set by hardware when any entry of the VC Arbitration Table is written by software. This bit is Cleared by hardware when hardware finishes loading values stored in the VC Arbitration Table after software sets the Load VC Arbitration Table bit in the Port VC Control Register .</p> <p>Default value of this bit is 0b.</p> | RO |

7.9.1.6 VC Resource Capability Register §

The VC Resource Capability Register describes the capabilities and configuration of a particular Virtual Channel resource. § Figure 7-234 details allocation of register fields in the VC Resource Capability Register ; § Table 7-213 provides the respective bit definitions.

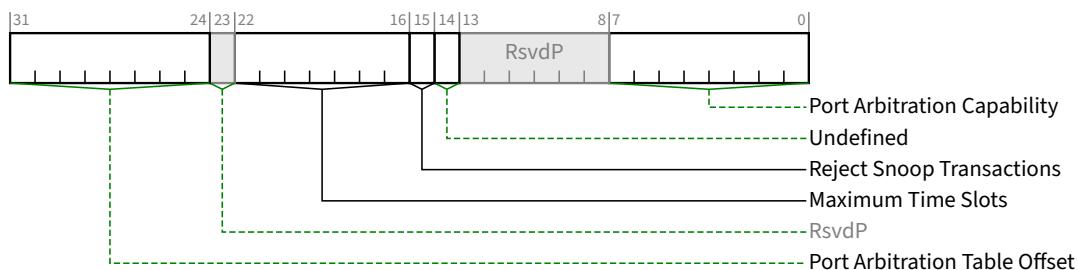


Figure 7-234 VC Resource Capability Register §

Table 7-213 VC Resource Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | <p>Port Arbitration Capability - Indicates types of Port Arbitration supported by the VC resource. This field is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs , but not for Endpoints or Root Ports that do not support peer-to-peer traffic.</p> <p>Each Bit Location within this field corresponds to a Port Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the VC resource can be configured to provide different arbitration services.</p> | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>Software selects among these capabilities by writing to the <u>Port Arbitration Select</u> field (see § <u>Section 7.9.1.7</u>).</p> <p>Defined bit positions are:</p> <ul style="list-style-type: none"> Bit 0 Non-configurable hardware-fixed arbitration scheme, e.g., Round Robin (RR) Bit 1 Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 WRR arbitration with 64 phases Bit 3 WRR arbitration with 128 phases Bit 4 Time-based WRR with 128 phases Bit 5 WRR arbitration with 256 phases Bits 6-7 Reserved | |
| 14 | Undefined Undefined - The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate Advanced Packet Switching. System software must ignore the value read from this bit. | RO |
| 15 | Reject Snoop Transactions - When Clear, transactions with or without the No Snoop bit Set within the TLP header are allowed on this VC. When Set, any transaction for which the No Snoop attribute is applicable but is not Set within the TLP header is permitted to be rejected as an Unsupported Request. Refer to § <u>Section 2.2.6.5</u> for information on where the No Snoop attribute is applicable. This bit is valid for Root Ports and RCRB ; it is not valid for Endpoints or Switch Ports. | HwInit |
| 22:16 | Maximum Time Slots - Indicates the maximum number of time slots (minus one) that the VC resource is capable of supporting when it is configured for time-based WRR Port Arbitration. For example, a value 000 0000b in this field indicates the supported maximum number of time slots is 1 and a value of 111 1111b indicates the supported maximum number of time slots is 128. This field is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs , but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. In addition, this field is valid only when the <u>Port Arbitration Capability</u> field indicates that the VC resource supports time-based WRR Port Arbitration. | HwInit |
| 31:24 | <p>Port Arbitration Table Offset - Indicates the location of the Port Arbitration Table associated with the VC resource. This field is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs , but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic.</p> <p>This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the <u>Virtual Channel Extended Capability</u> structure. A value of 00h indicates that the table is not present.</p> | RO |

7.9.1.7 VC Resource Control Register §

§ Figure 7-235 details allocation of register fields in the VC Resource Control Register ; § Table 7-214 provides the respective bit definitions.

Base 6.4 vs Base 6.3

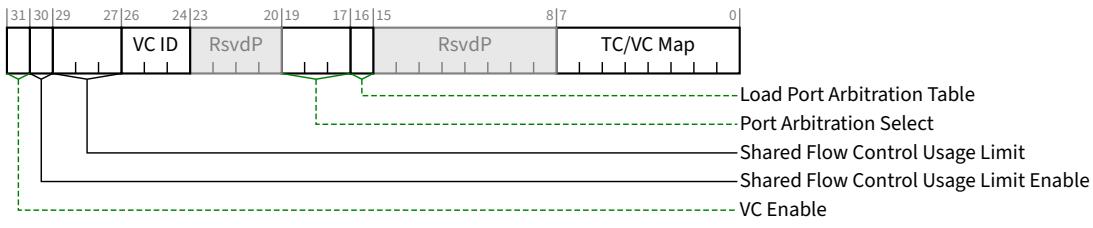


Figure 7-235 VC Resource Control Register §

Table 7-214 VC Resource Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------------------------|
| 7:0 | <p>TC/VC Map - This field indicates the TCs that are mapped to the VC resource. This field is valid for all Functions.</p> <p>Bit locations within this field correspond to TC values. For example, when bit 7 is Set in this field, TC7 is mapped to this VC resource. When more than 1 bit in this field is Set, it indicates that multiple TCs are mapped to the VC resource.</p> <p>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.</p> <p>Default value of this field is FFh for the first VC resource and is 00h for other VC resources.</p> <p>Note:</p> <p>Bit 0 of this field is read-only. It must be Set for the default VC0 and Clear for all other enabled VCs.</p> | RW (see the note for exceptions) |
| 16 | <p>Load Port Arbitration Table - When Set, this bit updates the Port Arbitration logic from the Port Arbitration Table for the VC resource. This bit is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs , but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. In addition, this bit is only valid when the Port Arbitration Table is used by the selected Port Arbitration scheme (that is indicated by a Set bit in the Port Arbitration Capability field selected by Port Arbitration Select).</p> <p>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.</p> <p>This bit always returns 0b when read.</p> <p>Default value of this bit is 0b.</p> | RW |
| 19:17 | <p>Port Arbitration Select - This field configures the VC resource to provide a particular Port Arbitration service. This field is valid for RCRBs , Root Ports that support peer-to-peer traffic, and Switch Ports, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic.</p> <p>The permissible value of this field is a number corresponding to one of the asserted bits in the Port Arbitration Capability field of the VC resource.</p> | RW |
| 26:24 | <p>VC ID - This field assigns a VC ID to the VC resource (see note for exceptions). This field is valid for all Functions.</p> <p>This field cannot be modified when the VC is already enabled.</p> <p>Note:</p> <p>For the first VC resource (default VC), this field is read-only and must be hardwired to 000b.</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|---|------------------------|----|-------------|-------|-------------|-----|-------------|-------|-------------|-----|-------------|-------|-------------|-----|-------------|-------|------------------------|
| 29:27 | <p>Shared Flow Control Usage Limit - this field controls what percentage of the available Shared Flow Control a given FC/VC is permitted to consume.</p> <p>This limit is applied independently for each Flow Control credit type. For example, if this field contains 101b and Shared Flow Control Usage Limit Enable is Set, a Posted TLP may not pass the Tx Gate if doing so would cause that VC to consume more than 62.5% of the available Shared Posted Header credits or if doing so would cause that VC to consume more than 62.5% of the available Shared Data credits.</p> <p>If Shared Flow Control Usage Limit Enable is Clear, this field is ignored and this VC is permitted to consume all of the shared credits.</p> <p>When Shared Flow Control Usage Limit Enable is Set, and this field contains 000b, this VC is not permitted to consume any shared credits.</p> <p>Behavior is undefined when all VCs have Shared Flow Control Usage Limit Enable Set and the sum of the Shared Flow Control Limit values for all VCs is less than 100%.</p> <p>Encodings are:</p> <table> <tbody> <tr><td>000b</td><td>0%</td></tr> <tr><td>001b</td><td>12.5%</td></tr> <tr><td>010b</td><td>25%</td></tr> <tr><td>011b</td><td>37.5%</td></tr> <tr><td>100b</td><td>50%</td></tr> <tr><td>101b</td><td>62.5%</td></tr> <tr><td>110b</td><td>75%</td></tr> <tr><td>111b</td><td>87.5%</td></tr> </tbody> </table> <p>Behavior is undefined if this field changes value while VC Enable and Shared Flow Control Usage Limit Enable are both Set.</p> <p>This field is <u>RsvdP</u> when <u>Flit Mode Supported</u> is Clear.</p> <p>When <u>Extended VC Count</u> is 0, this field is permitted to be hardwired to any value.</p> <p>When this field is <u>RW</u>, the default value is implementation specific.</p> | 000b | 0% | 001b | 12.5% | 010b | 25% | 011b | 37.5% | 100b | 50% | 101b | 62.5% | 110b | 75% | 111b | 87.5% | <u>RW / RO / RsvdP</u> |
| 000b | 0% | | | | | | | | | | | | | | | | | |
| 001b | 12.5% | | | | | | | | | | | | | | | | | |
| 010b | 25% | | | | | | | | | | | | | | | | | |
| 011b | 37.5% | | | | | | | | | | | | | | | | | |
| 100b | 50% | | | | | | | | | | | | | | | | | |
| 101b | 62.5% | | | | | | | | | | | | | | | | | |
| 110b | 75% | | | | | | | | | | | | | | | | | |
| 111b | 87.5% | | | | | | | | | | | | | | | | | |
| 30 | <p>Shared Flow Control Usage Limit Enable - When Set, this bit enables use of the Shared Flow Control Usage Limit value above at the transmitter for this Virtual Channel.</p> <p>Behavior is undefined if the value of this bit changes while VC Enable is Set.</p> <p>This bit is <u>RsvdP</u> when <u>Flit Mode Supported</u> is Clear.</p> <p>When <u>Extended VC Count</u> is 0, this bit is permitted to be hardwired to 0b.</p> <p>When this bit is <u>RW</u>, the default value is implementation specific.</p> | <u>RW / RO / RsvdP</u> | | | | | | | | | | | | | | | | |
| 31 | <p>VC Enable - This bit, when Set, enables a Virtual Channel. The Virtual Channel is disabled when this bit is cleared. This bit is valid for all Functions.</p> <p>Software must use the VC Negotiation Pending bit to check whether the VC negotiation is complete.</p> <p>For VC0, the attribute is <u>RO</u>. If no SVC capability is implemented in this Port, this bit's value must be 1b; otherwise, this bit's value must always be the same value as the <u>Use VC/MFVC</u> bit in the <u>SVC Port Status Register</u>. See § <u>Section 6.3.5</u>.</p> <p>For other VCs, if no SVC capability is implemented in this Port or if the <u>Use VC/MFVC</u> bit is Set, the default value of this bit is 0b and the attribute is <u>RW</u>; otherwise, this bit must be <u>RO</u> with a value of 0b.</p> <p>To enable a Virtual Channel in a Port using VC mechanisms, the VC Enable bit for that Virtual Channel must be Set. The corresponding Virtual Channel in the Link partner Port must be enabled as well, and that Virtual Channel may be in SVC, VC, or MFVC capabilities. To disable a Virtual Channel, Virtual</p> | <u>RW / HwInit</u> | | | | | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>Channel must be disabled in both components on the Link. Software must ensure that no traffic is using a Virtual Channel at the time it is disabled. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.</p> <p>When this bit is forced to be RO with a value of 0b due to the Use VC/MFVC bit being Clear, its associated VC is disabled, rendering most of its control registers to be ineffective.</p> | |

7.9.1.8 VC Resource Status Register §

§ Figure 7-236 details allocation of register fields in the VC Resource Status Register ; § Table 7-215 provides the respective bit definitions.

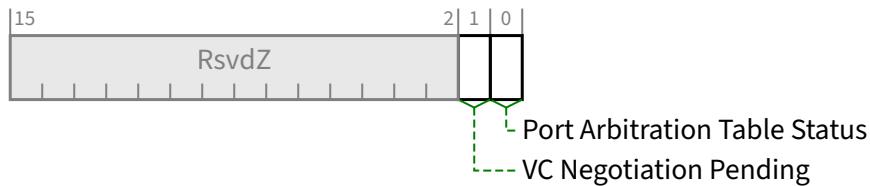


Figure 7-236 VC Resource Status Register §

Table 7-215 VC Resource Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Port Arbitration Table Status - This bit indicates the coherency status of the Port Arbitration Table associated with the VC resource. This bit is valid for RCRBs, Root Ports that support peer-to-peer traffic, and Switch Ports, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. In addition, this bit is valid only when the Port Arbitration Table is used by the selected Port Arbitration for the VC resource.</p> <p>This bit is Set by hardware when any entry of the Port Arbitration Table is written to by software. This bit is Cleared by hardware when hardware finishes loading values stored in the Port Arbitration Table after software sets the Load Port Arbitration Table bit.</p> <p>Default value of this bit is 0b.</p> | RO |
| 1 | <p>VC Negotiation Pending - This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state. This bit is valid for all Functions.</p> <p>The value of this bit is defined only when the Link is in the DL_Active state and the Virtual Channel is enabled (its VC Enable bit is Set).</p> <p>When this bit is Set by hardware, it indicates that the VC resource has not completed the process of negotiation. This bit is Cleared by hardware after the VC negotiation is complete (on exit from the FC_INIT2 state). For VC0, this bit is permitted to be hardwired to 0b.</p> <p>Before using a Virtual Channel, software must check whether the VC Negotiation Pending bits for that Virtual Channel are Clear in both components on the Link.</p> | RO |

7.9.1.9 VC Arbitration Table §

The VC Arbitration Table is a read-write register array that is used to store the arbitration table for VC Arbitration. This register array is valid for all Functions when the selected VC Arbitration uses a WRR table. Functions that do not support WRR VC arbitration are not required to implement a VC Arbitration Table. If it exists, the VC Arbitration Table is located by the VC Arbitration Table Offset field.

The VC Arbitration Table is a register array with fixed-size entries of 4 bits. § Figure 7-237 depicts the table structure of an example VC Arbitration Table with 32 phases. Each 4-bit table entry corresponds to a phase within a WRR arbitration period. The definition of table entry is depicted in § Table 7-216. The lower 3 bits (bits 0-2) contain the VC ID value, indicating that the corresponding phase within the WRR arbitration period is assigned to the Virtual Channel indicated by the VC ID (must be a valid VC ID that corresponds to an enabled VC).

The highest bit (bit 3) of the table entry is Reserved. The length of the table depends on the selected VC Arbitration as shown in § Table 7-217.

When the VC Arbitration Table is used by the default VC Arbitration method, the default values of the table entries must be all zero to ensure forward progress for the default VC (with VC ID of 0).

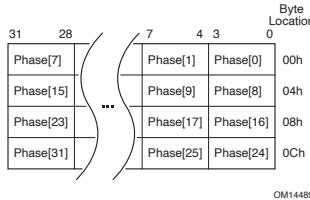


Figure 7-237 Example VC Arbitration Table with 32 Phases §

Table 7-216 Definition of the 4-bit Entries in the VC Arbitration Table §

| Bit Location | Description | Attributes |
|--------------|-------------|------------|
| 2:0 | VC ID | RW |
| 3 | RsvdP | RW |

Table 7-217 Length of the VC Arbitration Table §

| VC Arbitration Select | VC Arbitration Table Length |
|-----------------------|-----------------------------|
| 001b | 32 entries |
| 010b | 64 entries |
| 011b | 128 entries |

7.9.1.10 Port Arbitration Table §

The Port Arbitration Table register is a read-write register array that is used to store the WRR or time-based WRR arbitration table for Port Arbitration for the VC resource. This register array is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. It is only present when one or more asserted bits in the Port Arbitration Capability field indicate that the component

supports a Port Arbitration scheme that uses a programmable arbitration table. Furthermore, it is only valid when one of the above-mentioned bits in the Port Arbitration Capability field is selected by the Port Arbitration Select field.

The Port Arbitration Table represents one Port arbitration period. § Figure 7-238 shows the structure of an example Port Arbitration Table with 128 phases and 2-bit table entries. Each table entry containing a Port Number corresponds to a phase within a Port arbitration period. For example, a table with 2-bit entries can be used by a Switch component with up to four Ports. A Port Number written to a table entry indicates that the phase within the Port Arbitration period is assigned to the selected PCI Express Port (the Port Number must be a valid one).

- When the WRR Port Arbitration is used for a VC of any Egress Port, at each arbitration phase, the Port Arbiter serves one transaction from the Ingress Port indicated by the Port Number of the current phase. When finished, it immediately advances to the next phase. A phase is skipped, i.e., the Port Arbiter simply moves to the next phase immediately if the Ingress Port indicated by the phase does not contain any transaction for the VC (note that a phase cannot contain the Egress Port's Port Number).
- When the Time-based WRR Port Arbitration is used for a VC of any given Port, at each arbitration phase aligning to a virtual timeslot, the Port Arbiter serves one transaction from the Ingress Port indicated by the Port Number of the current phase. It advances to the next phase at the next virtual timeslot. A phase indicates an “idle” timeslot, i.e., the Port Arbiter does not serve any transaction during the phase, if:
 - the phase contains the Egress Port's Port Number, or
 - the Ingress Port indicated by the phase does not contain any transaction for the VC.
 - The Port Arbitration Table Entry Size field in the Port VC Capability Register 1 determines the table entry size. The length of the table is determined by the Port Arbitration Select field as shown in § Table 7-218 .
 - When the Port Arbitration Table is used by the default Port Arbitration for the default VC, the default values for the table entries must contain at least one entry for each of the other PCI Express Ports of the component to ensure forward progress for the default VC for each Port. The table may contain RR or RR-like fair Port Arbitration for the default VC.

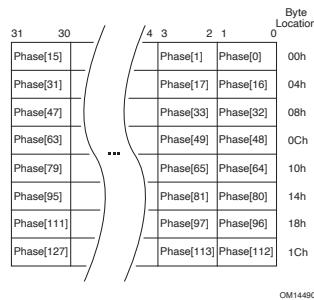


Figure 7-238 Example Port Arbitration Table with 128 Phases and 2-bit Table Entries §

Table 7-218 Length of Port Arbitration Table §

| Port Arbitration Select | Port Arbitration Table Length |
|-------------------------|-------------------------------|
| 001b | 32 entries |
| 010b | 64 entries |
| 011b | 128 entries |
| 100b | 128 entries |
| 101b | 256 entries |

7.9.2 Multi-Function Virtual Channel Extended Capability §

The Multi-Function Virtual Channel Extended Capability (**MFVC Capability**) is an optional Extended Capability that permits enhanced QoS management in a Multi-Function Device, including TC/VC mapping, optional VC arbitration, and optional Function arbitration for Upstream Requests. When implemented, the MFVC Extended Capability structure must be present in the Extended Configuration Space of Function 0 of the Multi-Function Device's Upstream Port. § Figure 7-239 provides a high level view of the MFVC Extended Capability structure. This MFVC Extended Capability structure controls Virtual Channel assignment at the PCI Express Upstream Port of the Multi-Function Device, while a VC Extended Capability structure, if present in a Function, controls the Virtual Channel assignment for that individual Function.

The number of (extended) Virtual Channels is indicated by the MFVC Extended VC Count field in the Port VC Capability Register 1. Software must interpret this field to determine the availability of extended MFVC VC Resource registers.

A Multi-Function Device is permitted to have an MFVC Extended Capability structure even if none of its Functions have a VC Extended Capability structure. However, an MFVC Extended Capability structure is permitted only in Function 0 in the Upstream Port of a Multi-Function Device.

↑↑The MFVC Extended Capability and the SIOV Extended Capability must not be in use at the same time. Software must not make use of the MFVC Extended Capability on a Multi-Function Device if SIOV is enabled on any of its PFs, otherwise the behavior is undefined. Similarly, if the MFVC Extended Capability is in use for a Multi-Function Device, software must not enable SIOV on any of its PFs, otherwise the behavior is undefined.↑

ECN: Base 6.3
SIOV△↔

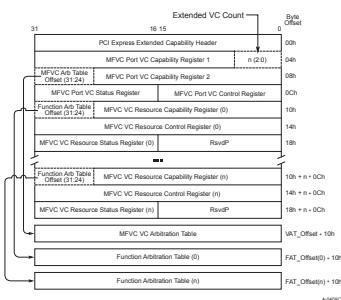


Figure 7-239 MFVC Capability Structure §

The following sections describe the registers/fields of the MFVC Extended Capability structure.

7.9.2.1 MFVC Extended Capability Header (Offset 00h) §

Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the MFVC Extended Capability is 0008h. § Figure 7-240 details allocation of register fields in the MFVC Extended Capability header; § Table 7-219 provides the respective bit definitions.

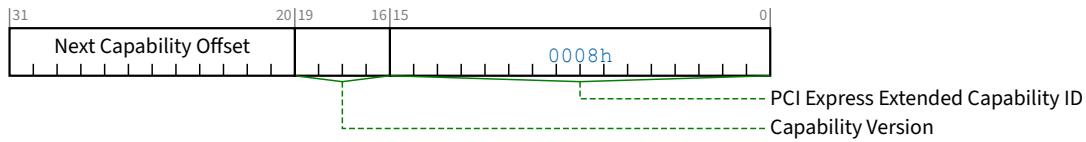


Figure 7-240 MFVC Extended Capability Header §

Table 7-219 MFVC Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the MFVC Extended Capability is 0008h. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.9.2.2 MFVC Port VC Capability Register 1 (Offset 04h) §

The MFVC Port VC Capability Register 1 describes the configuration of the Virtual Channels associated with a PCI Express Port of the Multi-Function Device . § Figure 7-241 details allocation of register fields in the MFVC Port VC Capability Register 1 ; § Table 7-220 provides the respective bit definitions.

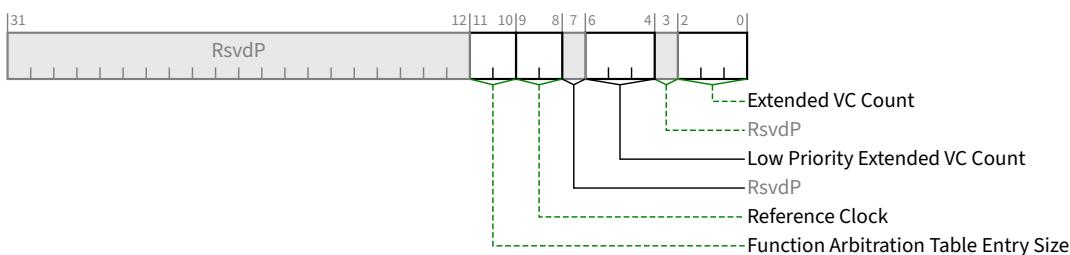


Figure 7-241 MFVC Port VC Capability Register 1 §

Base 6.4 vs Base 6.3

Table 7-220 MFVC Port VC Capability Register 1 §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>Extended VC Count - Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device.</p> <p>This value indicates the number of (extended) MFVC VC Resource Capability, Control, and Status registers that are present in Configuration Space in addition to the required MFVC VC Resource registers for the default VC.</p> <p>The minimum value of this field is 0 (for devices that only support the default VC and only have 1 set of MFVC VC Resource registers for that VC). The maximum value is 7.</p> | RO |
| 6:4 | <p>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.</p> <p>The minimum value of this field is 000b and the maximum value is Extended VC Count .</p> | RO |
| 9:8 | <p>Reference Clock - Indicates the reference clock for Virtual Channels that support time-based WRR Function Arbitration.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b 100 ns reference clock 01b - 11b Reserved | RO |
| 11:10 | <p>Function Arbitration Table Entry Size - Indicates the size (in bits) of Function Arbitration table entry in the device.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b Size of Function Arbitration table entry is 1 bit 01b Size of Function Arbitration table entry is 2 bits 10b Size of Function Arbitration table entry is 4 bits 11b Size of Function Arbitration table entry is 8 bits | RO |

7.9.2.3 MFVC Port VC Capability Register 2 (Offset 08h) §

The MFVC Port VC Capability Register 2 provides further information about the configuration of the Virtual Channels associated with a PCI Express Port of the Multi-Function Device . § Figure 7-242 details allocation of register fields in the MFVC Port VC Capability Register 2 ; § Table 7-221 provides the respective bit definitions.

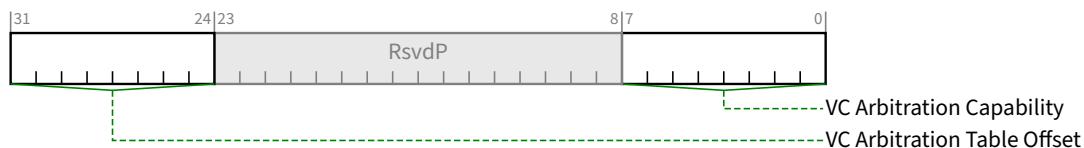


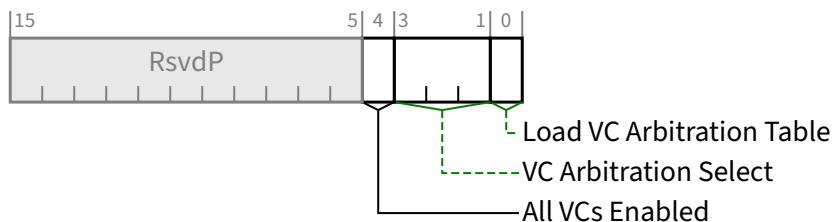
Figure 7-242 MFVC Port VC Capability Register 2 §

Table 7-221 MFVC Port VC Capability Register 2

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | <p>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.</p> <p>Each Bit Location within this field corresponds to a VC Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the device can be configured to provide different VC arbitration services.</p> <p>Defined bit positions are:</p> <ul style="list-style-type: none"> Bit 0 Hardware fixed arbitration scheme, e.g., Round Robin Bit 1 Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 WRR arbitration with 64 phases Bit 3 WRR arbitration with 128 phases Bits 4-7 Reserved | RO |
| 31:24 | <p>VC Arbitration Table Offset - Indicates the location of the <u>MFVC VC Arbitration Table</u>.</p> <p>This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the MFVC Extended Capability structure. A value of 00h indicates that the table is not present.</p> | RO |

7.9.2.4 MFVC Port VC Control Register (Offset 0Ch)

§ Figure 7-243 details allocation of register fields in the Port VC Control register; § Table 7-222 provides the respective bit definitions.

*Figure 7-243 MFVC Port VC Control Register**Table 7-222 MFVC Port VC Control Register*

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Load VC Arbitration Table - Used by software to update the <u>MFVC VC Arbitration Table</u>. This bit is valid when the selected VC Arbitration uses the <u>MFVC VC Arbitration Table</u>.</p> <p>Software Sets this bit to request hardware to apply new values programmed into <u>MFVC VC Arbitration Table</u>; Clearing this bit has no effect. Software checks the VC Arbitration Table Status bit in the MFVC Port VC Status register to confirm that new values stored in the <u>MFVC VC Arbitration Table</u> are latched by the VC arbitration logic.</p> <p>This bit always returns 0b when read.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 3:1 | <p>VC Arbitration Select - Used by software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration Capability field in the MFVC Port VC Capability Register 2.</p> <p>The permissible values of this field are numbers corresponding to one of the asserted bits in the VC Arbitration Capability field.</p> <p>This field cannot be modified when more than one VC in the LPVC group is enabled.</p> | RW |
| 4 | <p>All VCs Enabled - Setting this bit indicates that all VCs that will be used by the Port have been enabled. Setting this bit allows hardware to allocate assigned buffer resources across the enabled VCs.</p> <p>Setting this bit is optional. If this bit remains Clear and some VC Resources are never enabled, performance may be affected but the Link and all enabled VCs must operate correctly.</p> <p>Behavior is undefined if this bit is Set and any VC Enable bit in this capability changes value.</p> <p>Default value of this bit is 0b.</p> | RW |

7.9.2.5 MFVC Port VC Status Register (Offset 0Eh) §

The MFVC Port VC Status Register provides status of the configuration of Virtual Channels associated with a Port of the Multi-Function Device . § Figure 7-244 details allocation of register fields in the MFVC Port VC Status Register ; § Table 7-223 provides the respective bit definitions.

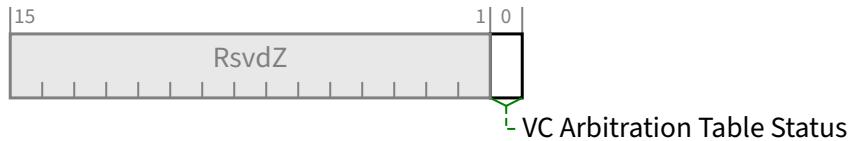


Figure 7-244 MFVC Port VC Status Register §

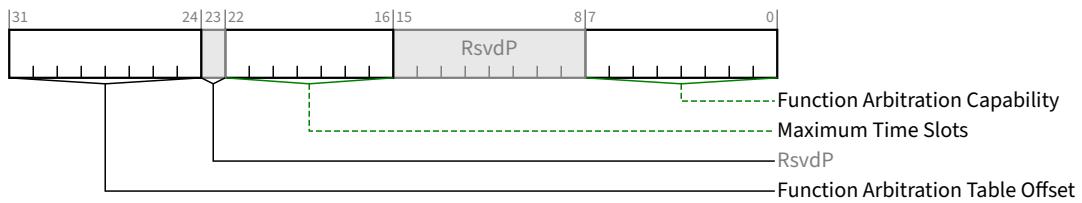
Table 7-223 MFVC Port VC Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>VC Arbitration Table Status - Indicates the coherency status of the MFVC VC Arbitration Table . This bit is valid when the selected VC uses the MFVC VC Arbitration Table .</p> <p>This bit is Set by hardware when any entry of the MFVC VC Arbitration Table is written by software. This bit is Cleared by hardware when hardware finishes loading values stored in the MFVC VC Arbitration Table after software sets the Load VC Arbitration Table bit in the MFVC Port VC Control Register .</p> <p>Default value of this bit is 0b.</p> | RO |

7.9.2.6 MFVC VC Resource Capability Register §

The MFVC VC Resource Capability Register describes the capabilities and configuration of a particular Virtual Channel resource. § Figure 7-245 details allocation of register fields in the MFVC VC Resource Capability Register ; § Table 7-224 provides the respective bit definitions.

Base 6.4 vs Base 6.3

Figure 7-245 MFVC VC Resource Capability Register [§](#)Table 7-224 MFVC VC Resource Capability Register [§](#)

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | <p>Function Arbitration Capability - Indicates types of Function Arbitration supported by the VC resource. Each Bit Location within this field corresponds to a Function Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the VC resource can be configured to provide different arbitration services.</p> <p>Software selects among these capabilities by writing to the Function Arbitration Select field (see § Section 7.9.2.7).</p> <p>Defined bit positions are:</p> <ul style="list-style-type: none"> Bit 0 Non-configurable hardware-fixed arbitration scheme, e.g., Round Robin (RR) Bit 1 Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 WRR arbitration with 64 phases Bit 3 WRR arbitration with 128 phases Bit 4 Time-based WRR with 128 phases Bit 5 WRR arbitration with 256 phases Bits 6-7 Reserved | RO |
| 22:16 | <p>Maximum Time Slots - Indicates the maximum number of time slots (minus 1) that the VC resource is capable of supporting when it is configured for time-based WRR Function Arbitration. For example, a value of 000 0000b in this field indicates the supported maximum number of time slots is 1 and a value of 111 1111b indicates the supported maximum number of time slots is 128.</p> <p>This field is valid only when the Function Arbitration Capability indicates that the VC resource supports time-based WRR Function Arbitration.</p> | HwInit |
| 31:24 | <p>Function Arbitration Table Offset - Indicates the location of the Function Arbitration Table associated with the VC resource.</p> <p>This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the MFVC Extended Capability structure. A value of 00h indicates that the table is not present.</p> | RO |

7.9.2.7 MFVC VC Resource Control Register [§](#)

[§ Figure 7-246 details allocation of register fields in the MFVC VC Resource Control Register](#); [§ Table 7-225 provides the respective bit definitions](#).

Base 6.4 vs Base 6.3

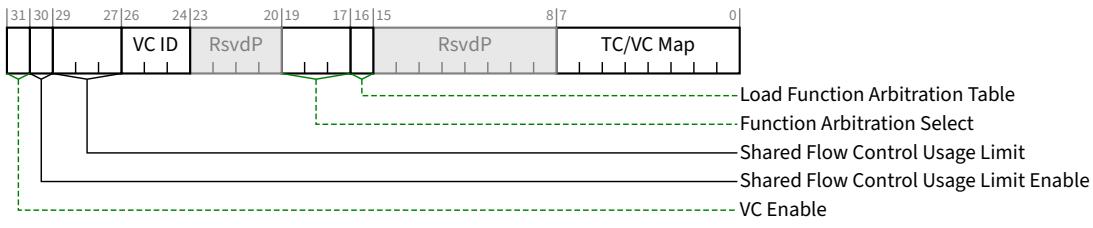


Figure 7-246 MFVC VC Resource Control Register §

Table 7-225 MFVC VC Resource Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------------------|
| 7:0 | <p>TC/VC Map - This field indicates the TCs that are mapped to the VC resource.</p> <p>Bit Locations within this field correspond to TC values. For example, when bit 7 is Set in this field, TC7 is mapped to this VC resource. When more than 1 bit in this field is Set, it indicates that multiple TCs are mapped to the VC resource.</p> <p>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.</p> <p>Default value of this field is FFh for the first VC resource and is 00h for other VC resources.</p> <p>Note:</p> <p>Bit 0 of this field is read-only. It must be hardwired to 1b for the default VC0 and hardwired to 0b for all other enabled VCs.</p> | RW (see the note for exceptions) |
| 16 | <p>Load Function Arbitration Table - When Set, this bit updates the Function Arbitration logic from the Function Arbitration Table for the VC resource. This bit is only valid when the Function Arbitration Table is used by the selected Function Arbitration scheme (that is indicated by a Set bit in the Function Arbitration Capability field selected by Function Arbitration Select).</p> <p>Software sets this bit to signal hardware to update Function Arbitration logic with new values stored in the Function Arbitration Table; clearing this bit has no effect. Software uses the Function Arbitration Table Status bit to confirm whether the new values of Function Arbitration Table are completely latched by the arbitration logic.</p> <p>This bit always returns 0b when read.</p> <p>Default value of this bit is 0b.</p> | RW |
| 19:17 | <p>Function Arbitration Select - This field configures the VC resource to provide a particular Function Arbitration service.</p> <p>The permissible value of this field is a number corresponding to one of the asserted bits in the Function Arbitration Capability field of the VC resource.</p> | RW |
| 26:24 | <p>VC ID - This field assigns a VC ID to the VC resource (see note for exceptions).</p> <p>This field cannot be modified when the VC is already enabled.</p> <p>Note:</p> <p>For the first VC resource (default VC), this field is a read-only field that must be hardwired to 000b.</p> | RW |
| 29:27 | <p>Shared Flow Control Usage Limit - this field controls what percentage of the available Shared Flow Control a given FC/VC is permitted to consume.</p> | RW / RO / RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|--------------|--|------------------------|----|-------------|-------|-------------|-----|-------------|-------|-------------|-----|-------------|-------|-------------|-----|-------------|-------|--|
| | <p>This limit is applied independently for each Flow Control credit type. For example, if this field contains 101b and Shared Flow Control Usage Limit Enable is Set, a Posted TLP may not pass the Tx Gate if doing so would cause that VC to consume more than 62.5% of the available Shared Posted Header credits or if doing so would cause that VC to consume more than 62.5% of the available Shared Data credits.</p> <p>If Shared Flow Control Usage Limit Enable is Clear, this field is ignored and this VC is permitted to consume all of the shared credits.</p> <p>When Shared Flow Control Usage Limit Enable is Set, and this field contains 000b, this VC is not permitted to consume any shared credits.</p> <p>Behavior is undefined when all VCs have Shared Flow Control Usage Limit Enable Set and the sum of the Shared Flow Control Limit values for all VCs is less than 100%.</p> <p>Encodings are:</p> <table> <tbody> <tr><td>000b</td><td>0%</td></tr> <tr><td>001b</td><td>12.5%</td></tr> <tr><td>010b</td><td>25%</td></tr> <tr><td>011b</td><td>37.5%</td></tr> <tr><td>100b</td><td>50%</td></tr> <tr><td>101b</td><td>62.5%</td></tr> <tr><td>110b</td><td>75%</td></tr> <tr><td>111b</td><td>87.5%</td></tr> </tbody> </table> <p>Behavior is undefined if this field changes value while VC Enable and Shared Flow Control Usage Limit Enable are both Set.</p> <p>This field is <u>RsvdP</u> when <u>Flit Mode Supported</u> is Clear.</p> <p>When <u>Extended VC Count</u> is 0, this field is permitted to be hardwired to 000b.</p> <p>When this field is <u>RW</u>, the default value is implementation specific.</p> | 000b | 0% | 001b | 12.5% | 010b | 25% | 011b | 37.5% | 100b | 50% | 101b | 62.5% | 110b | 75% | 111b | 87.5% | |
| 000b | 0% | | | | | | | | | | | | | | | | | |
| 001b | 12.5% | | | | | | | | | | | | | | | | | |
| 010b | 25% | | | | | | | | | | | | | | | | | |
| 011b | 37.5% | | | | | | | | | | | | | | | | | |
| 100b | 50% | | | | | | | | | | | | | | | | | |
| 101b | 62.5% | | | | | | | | | | | | | | | | | |
| 110b | 75% | | | | | | | | | | | | | | | | | |
| 111b | 87.5% | | | | | | | | | | | | | | | | | |
| 30 | <p>Shared Flow Control Usage Limit Enable - When Set, this bit enables use of the Shared Flow Control Usage Limit value above at the transmitter for this Virtual Channel.</p> <p>Behavior is undefined if the value of this bit changes while VC Enable is Set.</p> <p>This bit is <u>RsvdP</u> when <u>Flit Mode Supported</u> is Clear.</p> <p>When <u>Extended VC Count</u> is 0, this bit is permitted to be hardwired to 0b.</p> <p>When this bit is <u>RW</u>, the default value is implementation specific.</p> | <u>RW / RO / RsvdP</u> | | | | | | | | | | | | | | | | |
| 31 | <p>VC Enable - When Set, this bit enables a Virtual Channel. The Virtual Channel is disabled when this bit is cleared.</p> <p>Software must use the VC Negotiation Pending bit to check whether the VC negotiation is complete.</p> <p>For VC0, the attribute is <u>RO</u>. If no SVC capability is implemented in this Port, this bit's value must be 1b; otherwise, this bit's value must always be the same value as the <u>Use VC/MFVC</u> bit in the <u>SVC Port Status Register</u>. See § Section 6.3.5.</p> <p>For other VCs, if no SVC capability is implemented in this Port or if the <u>Use VC/MFVC</u> bit is Set, the default value of this bit is 0b and the attribute is <u>RW</u>; otherwise, this bit must be <u>RO</u> with a value of 0b.</p> <p>To enable a Virtual Channel, in a Port using MFVC mechanisms, the VC Enable bit for that Virtual Channel must be Set. The corresponding Virtual Channel in the Link partner Port must be enabled as well, and that Virtual Channel may be in SVC, VC, or MFVC capabilities. To disable a Virtual Channel, the Virtual Channel must be disabled in both components on the Link. Software must ensure that no traffic</p> | <u>RW / HwInit</u> | | | | | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>is using a Virtual Channel at the time it is disabled. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.</p> <p>When this bit is forced to be RO with a value of 0b due to the <u>Use VC/MFVC</u> bit being Clear, its associated VC is disabled, rendering most of its control registers to be ineffective.</p> | |

7.9.2.8 MFVC VC Resource Status Register §

§ Figure 7-247 details allocation of register fields in the MFVC VC Resource Status Register ; § Table 7-226 provides the respective bit definitions.

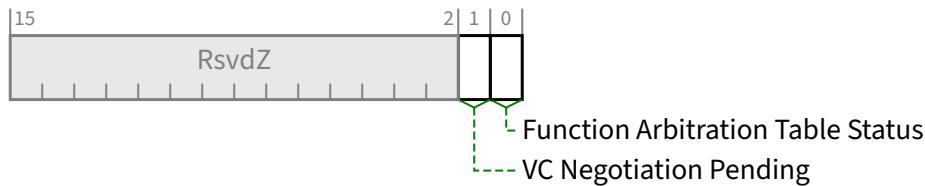


Figure 7-247 MFVC VC Resource Status Register §

Table 7-226 MFVC VC Resource Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Function Arbitration Table Status - This bit indicates the coherency status of the Function Arbitration Table associated with the VC resource. This bit is valid only when the Function Arbitration Table is used by the selected Function Arbitration for the VC resource.</p> <p>This bit is Set by hardware when any entry of the Function Arbitration Table is written to by software. This bit is Cleared by hardware when hardware finishes loading values stored in the Function Arbitration Table after software sets the Load Function Arbitration Table bit.</p> <p>Default value of this bit is 0b.</p> | RO |
| 1 | <p>VC Negotiation Pending - This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state.</p> <p>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.</p> <p>Before using a Virtual Channel, software must check whether the VC Negotiation Pending bits for that Virtual Channel are Clear in both components on a Link.</p> | RO |

7.9.2.9 MFVC VC Arbitration Table §

The definition of the MFVC VC Arbitration Table in the MFVC Extended Capability structure is identical to that in the VC Extended Capability structure (see § Section 7.9.1.9).

7.9.2.10 Function Arbitration Table §

The Function Arbitration Table register in the MFVC Extended Capability structure takes the same form as the Port Arbitration Table register in the VC Extended Capability structure (see § Section 7.9.1.10).

The Function Arbitration Table register is a read-write register array that is used to store the WRR or time-based WRR arbitration table for Function Arbitration for the VC resource. It is only present when one or more asserted bits in the Function Arbitration Capability field indicate that the Multi-Function Device supports a Function Arbitration scheme that uses a programmable arbitration table. Furthermore, it is only valid when one of the above-mentioned bits in the Function Arbitration Capability field is selected by the Function Arbitration Select field.

The Function Arbitration Table represents one Function arbitration period. Each table entry containing a Function Number or Function Group²⁰² Number corresponds to a phase within a Function Arbitration period. The table entry size requirements are as follows:

- The table entry size for non-ARI devices must support enough values to specify all implemented Functions plus at least one value that does not correspond to an implemented Function. For example, a table with 2-bit entries can be used by a Multi-Function Device with up to three Functions.
- The table entry size for ARI Devices must be either 4 bits or 8 bits.
 - If MFVC Function Groups are enabled, each entry maps to a single Function Group . Arbitration between multiple Functions within a Function Group is implementation specific, but must guarantee forward progress.
 - If MFVC Function Groups are not enabled and 4-bit entries are implemented, a given entry maps to all Functions whose Function Number modulo 8 matches its value. Similarly, if 8-bit entries are implemented, a given entry maps to all Functions whose Function Number modulo 128 matches its value. If a given entry maps to multiple Functions, arbitration between those Functions is implementation specific, but must guarantee forward progress.

A Function Number or Function Group Number written to a table entry indicates that the phase within the Function Arbitration period is assigned to the selected Function or Function Group (the Function Number or Function Group Number must be a valid one).

- When the WRR Function Arbitration is used for a VC of the Egress Port of the Multi-Function Device , at each arbitration phase the Function Arbiter serves one transaction from the Function or Function Group indicated by the Function Number or Function Group Number of the current phase. When finished, it immediately advances to the next phase. A phase is skipped, i.e., the Function Arbiter simply moves to the next phase immediately if the Function or Function Group indicated by the phase does not contain any transaction for the VC.
- When the Time-based WRR Function Arbitration is used for a VC of the Egress Port of the Multi-Function Device , at each arbitration phase aligning to a virtual timeslot, the Function Arbiter serves one transaction from the Function or Function Group indicated by the Function Number or Function Group Number of the current phase. It advances to the next phase at the next virtual timeslot. A phase indicates an “idle” timeslot, i.e., the Function Arbiter does not serve any transaction during the phase, if:
 - the phase contains the Number of a Function or a Function Group that does not exist, or
 - the Function or Function Group indicated by the phase does not contain any transaction for the VC.

The Function Arbitration Table Entry Size field in the MFVC Port VC Capability Register 1 determines the table entry size. The length of the table is determined by the Function Arbitration Select field as shown in § Table 7-227 .

202. If an ARI Device supports MFVC Function Groups capability and ARI-aware software enables it, arbitration is based on Function Groups instead of Functions. See § Section 7.8.8 .

When the Function Arbitration Table is used by the default Function Arbitration for the default VC, the default values for the table entries must contain at least one entry for each of the active Functions or Function Groups in the Multi-Function Device to ensure forward progress for the default VC for the Multi-Function Device's Upstream Port. The table may contain RR or RR-like fair Function Arbitration for the default VC.

Table 7-227 Length of Function Arbitration Table §

| Function Arbitration Select | Function Arbitration Table Length |
|-----------------------------|-----------------------------------|
| 001b | 32 entries |
| 010b | 64 entries |
| 011b | 128 entries |
| 100b | 128 entries |
| 101b | 256 entries |

7.9.3 Device Serial Number Extended Capability §

The Device Serial Number Extended Capability is an optional Extended Capability that may be implemented by any PCI Express device. § Figure 7-248 details allocation of register fields in the Device Serial Number Extended Capability structure.

It is permitted but not recommended for RCiEPs to implement this Capability.

RCiEPs that implement this Capability are permitted but not required to return the same Device Serial Number value as that reported by other RCiEPs of the same Root Complex.

All Multi-Function Devices other than RCiEPs that implement this Capability must implement it for Function 0; other Functions that implement this Capability must return the same Device Serial Number value as that reported by Function 0.

RCiEPs are permitted to implement or not implement this Capability on an individual basis, independent of whether they are part of a Multi-Function Device .

A PCI Express component other than a Root Complex containing multiple Devices such as a PCI Express Switch that implements this Capability must return the same Device Serial Number for each device.

The Device Serial Number Extended Capability is permitted to be present in PFs. If a PF contains the capability, its value applies to all associated VFs. VFs are permitted but not recommended to implement this capability. VFs that implement this capability must return the same Device Serial Number value as that reported by their associated PF.

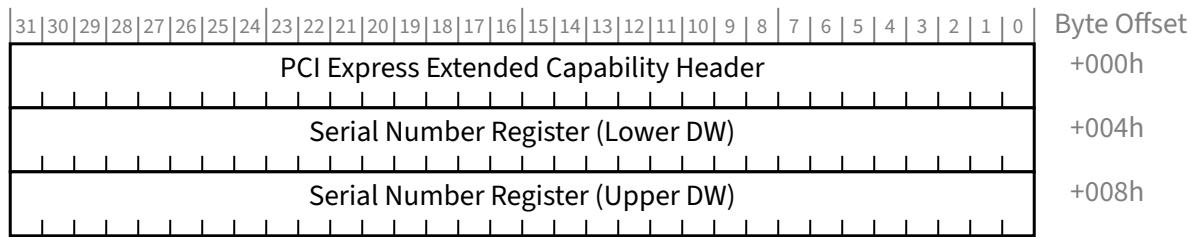


Figure 7-248 Device Serial Number Extended Capability Structure §

7.9.3.1 Device Serial Number Extended Capability Header (Offset 00h) §

§ Figure 7-249 details allocation of register fields in the Device Serial Number Extended Capability Header ; § Table 7-228 provides the respective bit definitions. Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Device Serial Number Extended Capability is 0003h .

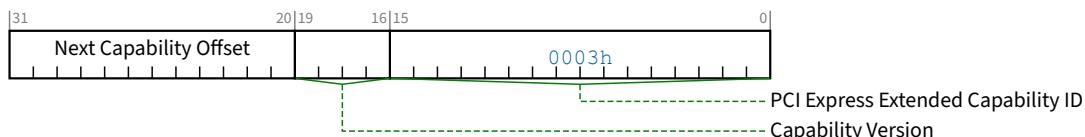


Figure 7-249 Device Serial Number Extended Capability Header §

Table 7-228 Device Serial Number Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Device Serial Number Extended Capability is 0003h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.9.3.2 Serial Number Register (Offset 04h) §

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit extended unique identifier [EUI-64]. § Figure 7-250 details allocation of register fields in the Serial Number register; § Table 7-229 provides the respective bit definitions.

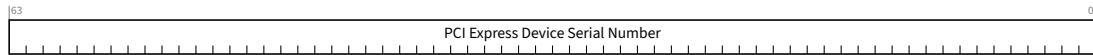


Figure 7-250 Serial Number Register §

Table 7-229 Serial Number Register §

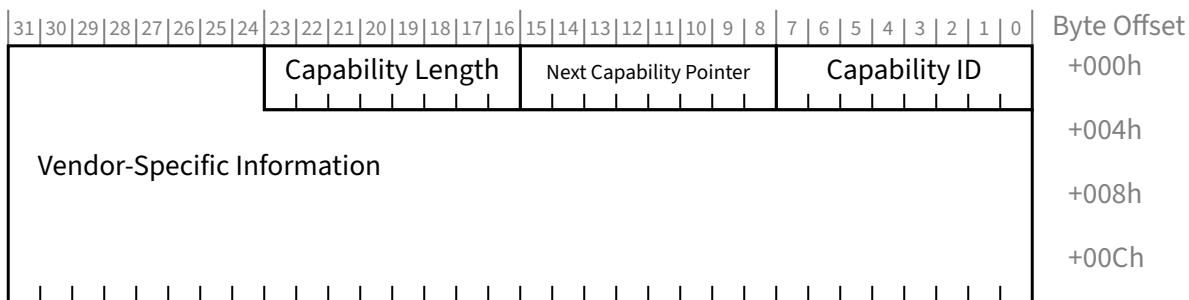
| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 63:0 | <p>PCI Express Device Serial Number - This field contains the IEEE defined 64-bit Extended Unique Identifier [EUI-64]. This identifier includes a 24-bit company id value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.</p> <p>PCI Express Device Serial Number[07:00] = EUI[63:56] PCI Express Device Serial Number[15:08] = EUI[55:48] PCI Express Device Serial Number[23:16] = EUI[47:40] PCI Express Device Serial Number[31:24] = EUI[39:32] PCI Express Device Serial Number[39:32] = EUI[31:24] PCI Express Device Serial Number[47:40] = EUI[23:16] PCI Express Device Serial Number[55:48] = EUI[15:08] PCI Express Device Serial Number[63:56] = EUI[07:00]</p> | RO |

7.9.4 Vendor-Specific Capability §

The Vendor-Specific Capability is a capability structure in PCI-compatible Configuration Space (first 256 bytes) as shown in § Figure 7-251 .

The Vendor-Specific Capability allows device vendors to use the Capability mechanism for vendor-specific information. The layout of the information is vendor-specific, except for the first three bytes, as explained below.

A single PCI Express Function is permitted to contain multiple VSEC structures.

Figure 7-251 *Vendor-Specific Capability* §Table 7-230 *Vendor-Specific Capability* §

| Bit Location | Register Description | Attributes |
|--------------|---|-----------------|
| 7:0 | Capability ID - Indicates the PCI Express Capability structure. This field must return a Capability ID of 09h indicating that this is a Vendor-Specific Capability structure. | RO |
| 15:8 | Next Capability Pointer - This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities. | RO |
| 23:16 | Capability Length - This field provides the number of bytes in the Capability structure (including the three bytes consumed by the Capability ID, Next Capability Pointer, and Capability Length field). | RO |
| 31:24 | Vendor Specific Information | Vendor Specific |

7.9.5 Vendor-Specific Extended Capability §

The Vendor-Specific Extended Capability (**VSEC Capability**) is an optional Extended Capability that is permitted to be implemented by any PCI Express Function or RCRB. This allows PCI Express component vendors to use the Extended Capability mechanism to expose vendor-specific registers.

A single PCI Express Function or RCRB is permitted to contain multiple VSEC structures.

An example usage is a set of vendor-specific features that are intended to go into an on-going series of components from that vendor. A VSEC structure can tell vendor-specific software which features a particular component supports, including components developed after the software was released.

§ Figure 7-252 details allocation of register fields in the VSEC structure. The structure of the Vendor-Specific Extended Capability Header and the Vendor-Specific Header is architected by this specification.

With a PCI Express Function, the structure and definition of the vendor-specific Registers area is determined by the vendor indicated by the Vendor ID field located at byte offset 00h in PCI-compatible Configuration Space. With an RCRB, a VSEC is permitted only if the RCRB also contains an RCRB Header Extended Capability structure, which contains a Vendor ID field indicating the vendor.

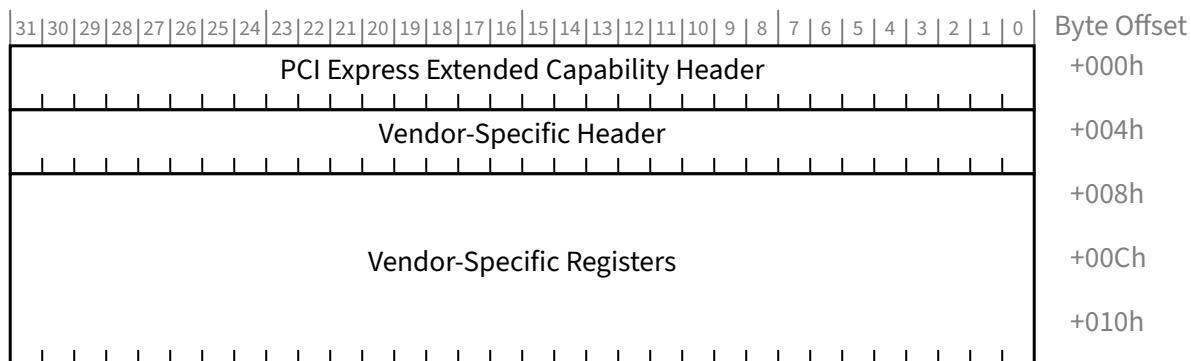


Figure 7-252 VSEC Capability Structure §

7.9.5.1 Vendor-Specific Extended Capability Header (Offset 00h) §

§ Figure 7-253 details allocation of register fields in the Vendor-Specific Extended Capability Header ; § Table 7-231 provides the respective bit definitions. Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability Header . The Extended Capability ID for the Vendor-Specific Extended Capability is 000Bh .

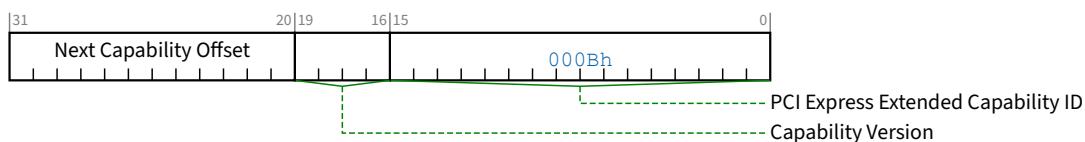


Figure 7-253 Vendor-Specific Extended Capability Header §

Table 7-231 Vendor-Specific Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Vendor-Specific Extended Capability is 000Bh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.9.5.2 Vendor-Specific Header (Offset 04h) §

§ Figure 7-254 details allocation of register fields in the Vendor-Specific Header ; § Table 7-232 provides the respective bit definitions.

Vendor-specific software must qualify the associated Vendor ID of the PCI Express Function or RCRB before attempting to interpret the values in the VSEC ID or VSEC Rev fields.

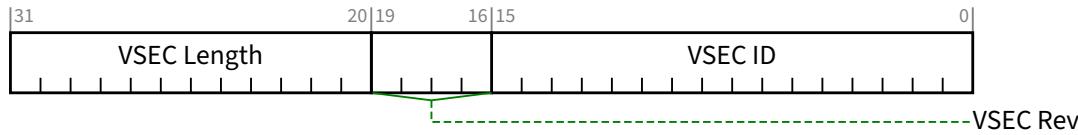


Figure 7-254 Vendor-Specific Header §

Table 7-232 Vendor-Specific Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | VSEC ID - This field is a vendor-defined ID number that indicates the nature and format of the VSEC structure. Software must qualify the Vendor ID before interpreting this field. | RO |
| 19:16 | VSEC Rev - This field is a vendor-defined version number that indicates the version of the VSEC structure. Software must qualify the Vendor ID and <u>VSEC ID</u> before interpreting this field. | RO |
| 31:20 | VSEC Length - This field indicates the number of bytes in the entire VSEC structure, including the <u>Vendor-Specific Extended Capability Header</u> , the <u>Vendor-Specific Header</u> , and the vendor-specific registers. | RO |

7.9.6 Designated Vendor-Specific Extended Capability (DVSEC) §

The Designated Vendor-Specific Extended Capability (**DVSEC Capability**) is an optional Extended Capability that is permitted to be implemented by any PCI Express Function or RCRB . This allows PCI Express component vendors to use the Extended Capability mechanism to expose vendor-specific registers that can be present in components by a variety of vendors.

A single PCI Express Function or RCRB is permitted to contain multiple DVSEC Capability structures.

An example usage is a set of vendor-specific features that are intended to go into an on-going series of components from a collection of vendors. A DVSEC Capability structure can tell vendor-specific software which features a particular component supports, including components developed after the software was released.

§ Figure 7-255 details allocation of register fields in the DVSEC Capability structure. The structure of the PCI Express Extended Capability Header and the Designated Vendor-Specific header is architected by this specification.

The DVSEC Vendor-Specific Register area begins at offset 0Ah.

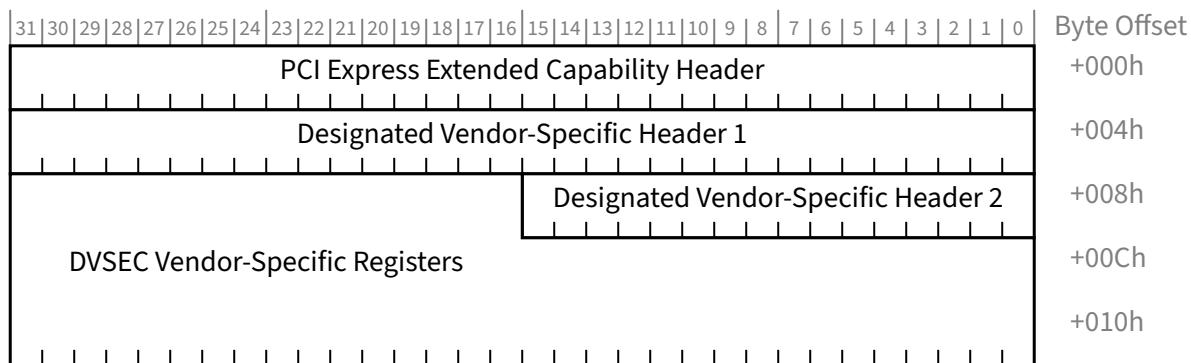


Figure 7-255 Designated Vendor-Specific Extended Capability §

7.9.6.1 Designated Vendor-Specific Extended Capability Header (Offset 00h) §

§ Figure 7-256 details allocation of register fields in the Designated Vendor-Specific Extended Capability Header ; § Table 7-233 provides the respective bit definitions. Refer to § Section 7.9.3 for a description of the PCI Express Extended Capability Header . The Extended Capability ID for the Designated Vendor-Specific Extended Capability is 0023h .

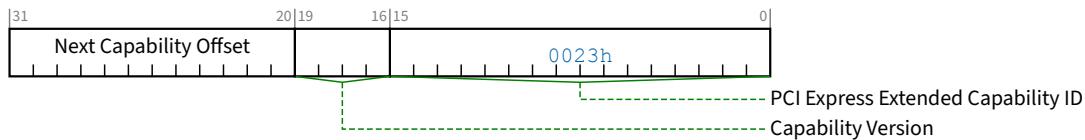


Figure 7-256 Designated Vendor-Specific Extended Capability Header §

Table 7-233 Designated Vendor-Specific Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Designated Vendor-Specific Extended Capability is 0023h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.9.6.2 Designated Vendor-Specific Header 1 (Offset 04h) §

§ Figure 7-257 details allocation of register fields in the Designated Vendor-Specific Header 1 ; § Table 7-234 provides the respective bit definitions.

Vendor-specific software must qualify the DVSEC Vendor ID before attempting to interpret the DVSEC Revision field.

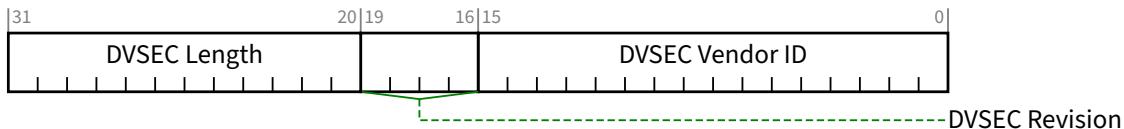


Figure 7-257 Designated Vendor-Specific Header 1 §

Table 7-234 Designated Vendor-Specific Header 1 §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | DVSEC Vendor ID - This field is the Vendor ID associated with the vendor that defined the contents of this capability. | RO |
| 19:16 | DVSEC Revision - This field is a vendor-defined version number that indicates the version of the DVSEC structure. Software must qualify the DVSEC Vendor ID and DVSEC ID before interpreting this field. | RO |
| 31:20 | DVSEC Length - This field indicates the number of bytes in the entire DVSEC structure, including the PCI Express Extended Capability Header , the DVSEC Header 1, DVSEC Header 2, and DVSEC vendor-specific registers. | RO |

7.9.6.3 Designated Vendor-Specific Header 2 (Offset 08h) §

§ Figure 7-258 details allocation of register fields in the Designated Vendor-Specific Header 2 ; § Table 7-235 provides the respective bit definitions.

Vendor-specific software must qualify the DVSEC Vendor ID before attempting to interpret the DVSEC ID field.



Figure 7-258 Designated Vendor-Specific Header 2 §

Table 7-235 Designated Vendor-Specific Header 2 §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | DVSEC ID - This field is a vendor-defined ID that indicates the nature and format of the DVSEC structure. Software must qualify the DVSEC Vendor ID before interpreting this field. | RO |

7.9.7 RCRB Header Extended Capability §

The PCI Express RCRB Header Extended Capability is an optional Extended Capability that may be implemented in an RCRB to provide a Vendor ID and Device ID for the RCRB and to permit the management of parameters that affect the behavior of Root Complex functionality associated with the RCRB .

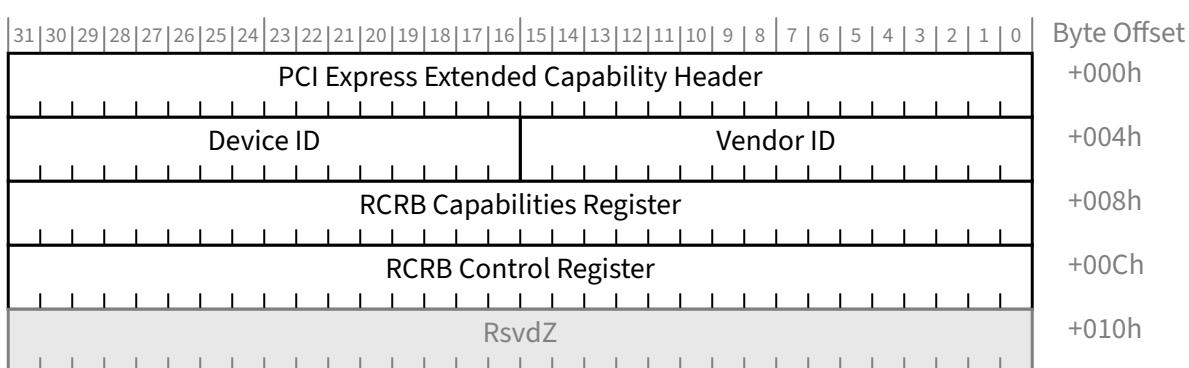


Figure 7-259 RCRB Header Extended Capability Structure §

7.9.7.1 RCRB Header Extended Capability Header (Offset 00h) §

§ Figure 7-260 details allocation of register fields in the RCRB Header Extended Capability Header . § Table 7-236 provides the respective bit definitions. Refer to § Section 7.6.3 for a description of the PCI Express Enhanced Capabilities header. The Extended Capability ID for the RCRB Header Extended Capability is 000Ah .

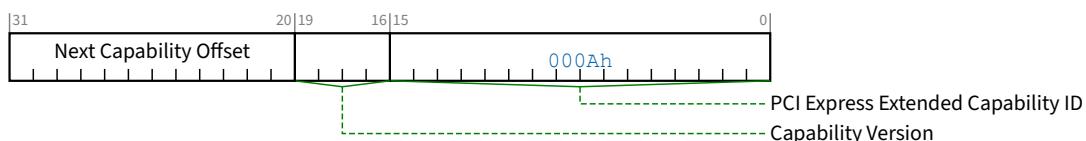


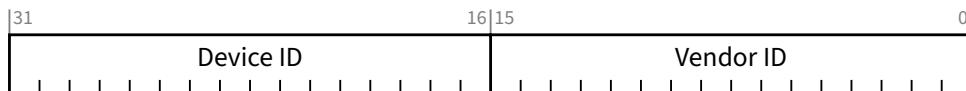
Figure 7-260 RCRB Header Extended Capability Header §

Table 7-236 RCRB Header Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the RCRB Header Extended Capability is 000Ah . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | RO |

7.9.7.2 RCRB Vendor ID and Device ID register (Offset 04h) §

§ Figure 7-261 details allocation of register fields in the RCRB Vendor ID and Device ID register ; § Table 7-237 provides the respective bit definitions.

*Figure 7-261 RCRB Vendor ID and Device ID register §**Table 7-237 RCRB Vendor ID and Device ID register §*

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Vendor ID - PCI-SIG assigned. Analogous to the equivalent field in PCI-compatible Configuration Space. This field provides a means to associate an RCRB with a particular vendor. | RO |
| 31:16 | Device ID - Vendor assigned. Analogous to the equivalent field in PCI-compatible Configuration Space. This field provides a means for a vendor to classify a particular RCRB . | RO |

7.9.7.3 RCRB Capabilities register (Offset 08h) §

§ Figure 7-262 details allocation of register fields in the RCRB Capabilities register ; § Table 7-238 provides the respective bit definitions.



Figure 7-262 RCRB Capabilities register §

Table 7-238 RCRB Capabilities register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Configuration RRS Software Visibility - When Set, this bit indicates that the Root Complex is capable of returning Request Retry Status (RRS) Completion Status in response to a Configuration Request for all Root Ports and integrated devices associated with this RCRB (see § Section 2.3.1). | RO |

7.9.7.4 RCRB Control register (Offset 0Ch) §

§ Figure 7-263 details allocation of register fields in the RCRB Control register ; § Table 7-239 provides the respective bit definitions.

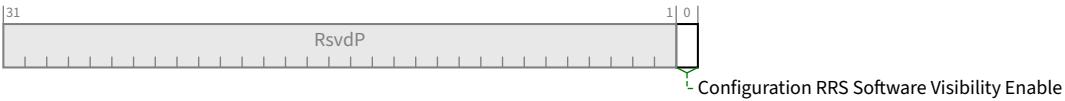


Figure 7-263 RCRB Control register §

Table 7-239 RCRB Control register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | Configuration RRS Software Visibility Enable - When Set, this bit enables the Root Complex to return Request Retry Status (RRS) Completion Status in response to a Configuration Reuest for all Root Ports and integrated devices associated with this RCRB (see § Section 2.3.1). RCRBs that do not implement this capability must hardwire this bit to 0b. Default value of this bit is 0b. | RW |

7.9.8 Root Complex Link Declaration Extended Capability §

The Root Complex Link Declaration Extended Capability is an optional Capability that is permitted to be implemented by Root Ports, RCiEPs , or RCRBs to declare a Root Complex's internal topology.

A Root Complex consists of one or more following elements:

- PCI Express Root Port
- A default system Egress Port or an internal sink unit such as memory (represented by an RCRB)
- Internal Data Paths/Links (represented by an RCRB on either side of an internal Link)

- Integrated devices
- Functions

A Root Complex Component is a logical aggregation of the above described Root Complex elements. No single element can be part of more than one Root Complex Component. Each Root Complex Component must have a unique Component ID.

A Root Complex is represented either as an opaque Root Complex or as a collection of one or more Root Complex Components.

The Root Complex Link Declaration Extended Capability is permitted to be present in a Root Complex element's Configuration Space or RCRB . It declares Links from the respective element to other elements of the same Root Complex Component or to an element in another Root Complex Component. The Links are required to be declared bidirectional such that each valid data path from one element to another has corresponding Link Entries in the Configuration Space (or RCRB) of both elements.

The Root Complex Link Declaration Extended Capability is permitted to also declare an association between a Configuration Space element (Root Port or RCiEP) and an RCRB Header Extended Capability (see § Section 7.9.7) contained in an RCRB that affects the behavior of the Configuration Space element. Note that an RCRB Header association is not declared bidirectional; the association is only declared by the Configuration Space element and not by the target RCRB .

IMPLEMENTATION NOTE: TOPOLOGIES TO AVOID §

Topologies that create more than one data path between any two Root Complex elements (either directly or through other Root Complex elements) may not be able to support bandwidth allocation in a standard manner. The description of how traffic is routed through such a topology is implementation specific, meaning that general purpose-operating systems may not have enough information about such a topology to correctly support bandwidth allocation. In order to circumvent this problem, these operating systems may require that a single RCRB element (of type Internal Link) not declare more than one Link to a Root Complex Component other than the one containing the RCRB element itself.

The Root Complex Link Declaration Extended Capability , as shown in § Figure 7-264 , consists of the PCI Express Extended Capability header and Root Complex Element Self Description followed by one or more Root Complex Link Entries .

Base 6.4 vs Base 6.3

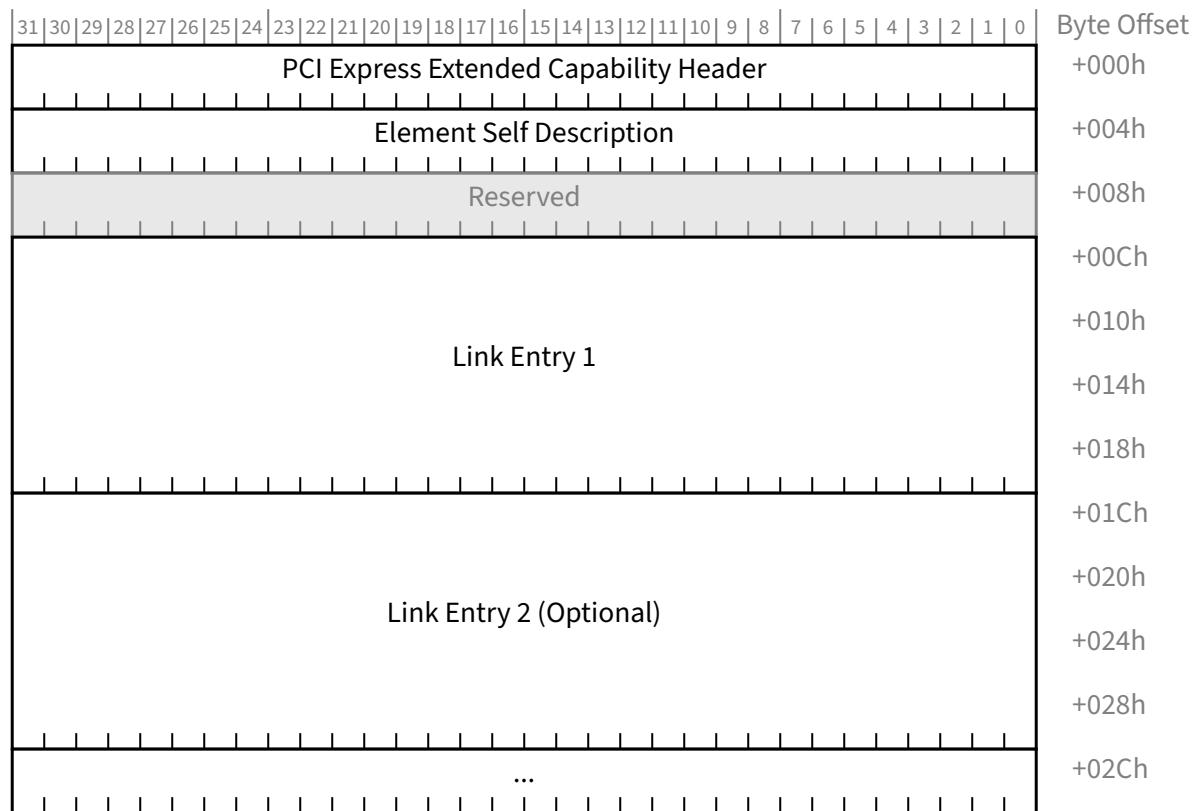


Figure 7-264 Root Complex Link Declaration Extended Capability §

7.9.8.1 Root Complex Link Declaration Extended Capability Header (Offset 00h) §

The Extended Capability ID for the Root Complex Link Declaration Extended Capability is 0005h .

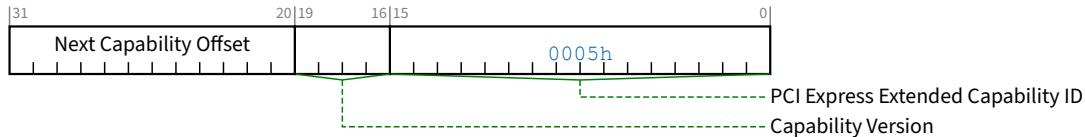


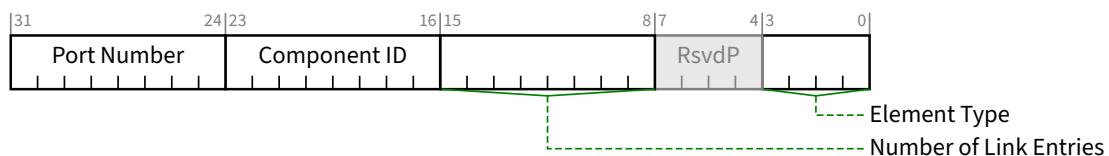
Figure 7-265 Root Complex Link Declaration Extended Capability Header §

Table 7-240 Root Complex Link Declaration Extended Capability Header §

| Bit Location ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|--|--|------------|
| 15:0 | <p>PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.</p> <p>The Extended Capability ID for the Root Complex Link Declaration Extended Capability is 0005h .</p> | RO |
| 19:16 | <p>Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.</p> <p>Must be 1h for this version of the specification.</p> | RO |
| 31:20 | <p>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.</p> <p>For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh.</p> <p>The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.</p> | RO |

7.9.8.2 Element Self Description Register (Offset 04h) §

The Element Self Description Register provides information about the Root Complex element containing the Root Complex Link Declaration Extended Capability .

*Figure 7-266 Element Self Description Register §**Table 7-241 Element Self Description Register §*

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | <p>Element Type - This field indicates the type of the Root Complex Element. Defined encodings are:</p> <ul style="list-style-type: none"> 0h Configuration Space Element 1h System Egress Port or internal sink (memory) 2h Internal Root Complex Link 3h-Fh Reserved | RO |
| 15:8 | Number of Link Entries - This field indicates the number of Link Entries following the Element Self Description. This field must report a value of 01h or higher. | HwInit |
| 23:16 | Component ID - This field identifies the Root Complex Component that contains this Root Complex Element. Component IDs must start at 01h, as a value of 00h is Reserved. | HwInit |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:24 | <p>Port Number - This field specifies the Port Number associated with this element with respect to the Root Complex Component that contains this element.</p> <p>An element with a Port Number of 00h indicates the default Egress Port to configuration software.</p> | HwInit |

7.9.8.3 Link Entries §

Link Entries start at offset 10h of the Root Complex Link Declaration Extended Capability structure. Each Link Entry consists of a Link description followed by a 64-bit Link Address at offset 08h from the start of Link Entry identifying the target element for the declared Link. A Link Entry declares an internal Link to another Root Complex Element.

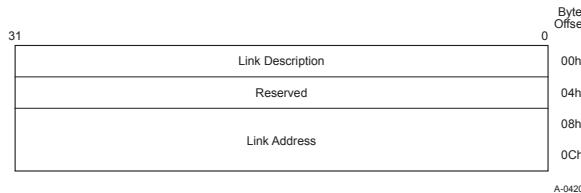


Figure 7-267 Link Entry §

7.9.8.3.1 Link Description Register §

The Link Description Register is located at offset 00h from the start of a Link Entry and is defined as follows:

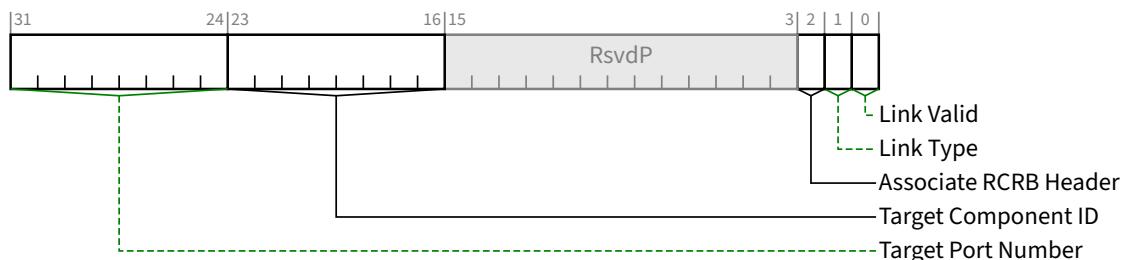


Figure 7-268 Link Description Register §

Table 7-242 Link Description Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Link Valid - When Set, this bit indicates that the Link Entry specifies a valid Link. Link Entries that do not have either this bit Set or the Associate RCRB Header bit Set (or both) are ignored by software.</p> | HwInit |
| 1 | <p>Link Type - This bit indicates the target type of the Link and defines the format of the Link Address field. Defined Link Type values are:</p> <ul style="list-style-type: none"> 0b Link points to memory-mapped space²⁰³ (for RCRB). The Link Address specifies the 64-bit base address of the target RCRB. | HwInit |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | 1b Link points to Configuration Space (for a Root Port or RCiEP). The Link Address specifies the configuration address (PCI Segment Group, Bus, Device, Function) of the target element. | |
| 2 | Associate RCRB Header - When Set, this bit indicates that the <u>Link Entry</u> associates the declaring element with an RCRB Header Extended Capability in the target RCRB. <u>Link Entries</u> that do not have either this bit Set or the <u>Link Valid</u> bit Set (or both) are ignored by software. The <u>Link Type</u> bit must be Clear when this bit is Set. | HwInit |
| 23:16 | Target Component ID - This field identifies the Root Complex Component that is targeted by this <u>Link Entry</u> . Components IDs must start at 01h, as a value of 00h is Reserved | HwInit |
| 31:24 | Target Port Number - This field specifies the Port Number associated with the element targeted by this <u>Link Entry</u> ; the <u>Target Port Number</u> is with respect to the Root Complex Component (identified by the <u>Target Component ID</u>) that contains the target element. | HwInit |

7.9.8.3.2 Link Address §

The Link Address is a HwInit field located at offset 08h from the start of a Link Entry that identifies the target element for the Link Entry. For a Link of Link Type 0 in its Link Description, the Link Address specifies the memory-mapped base address of RCRB. For a Link of Link Type 1 in its Link Description, the Link Address specifies the Configuration Space address of a PCI Express Root Port or an RCiEP.

7.9.8.3.2.1 Link Address for Link Type 0 §

For a Link pointing to a memory-mapped RCRB (Link Type bit = 0), the first DWORD specifies the lower 32 bits of the RCRB base address of the target element as shown below; bits 11:0 are hardwired to 000h and Reserved for future use. The second DWORD specifies the high order 32 bits (63:32) of the RCRB base address of the target element.

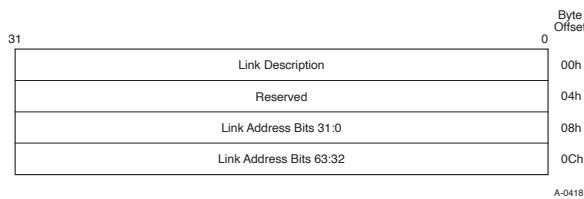


Figure 7-269 Link Address for Link Type 0 §

7.9.8.3.2.2 Link Address for Link Type 1 §

For a Link pointing to the Configuration Space of a Root Complex element (Link Type bit = 1), bits in the first DWORD specify the Bus, Device, and Function Number of the target element. As shown in § Figure 7-270, bits 2:0 (N) encode the number of bits n associated with the Bus Number, with N = 000b specifying n = 8 and all other encodings specifying n = <value of N>. Bits 11:3 are Reserved and hardwired to 0. Bits 14:12 specify the Function Number, and bits 19:15 specify the Device Number. Bits (19 + n):20 specify the Bus Number, with 1 ≤ n ≤ 8.

203. The memory-mapped space for accessing an RCRB is not the same as Memory Space, and must not overlap with Memory Space.

Bits 31:($20 + n$) of the first DWORD together with the second DWORD optionally identify the target element's hierarchy for systems implementing the PCI Express Enhanced Configuration Access Mechanism by specifying bits 63:($20 + n$) of the memory-mapped Configuration Space base address of the PCI Express hierarchy associated with the targeted element; single hierarchy systems that do not implement more than one memory mapped Configuration Space are allowed to report a value of zero to indicate default Configuration Space.

A Configuration Space base address [63:($20 + n$)] equal to zero indicates that the Configuration Space address defined by bits (19 + n):12 (Bus Number, Device Number, and Function Number) exists in the default PCI Segment Group; any non-zero value indicates a separate Configuration Space base address.

Software must not use n outside the context of evaluating the Bus Number and memory-mapped Configuration Space base address for this specific target element. In particular, n does not necessarily indicate the maximum Bus Number supported by the associated PCI Segment Group.

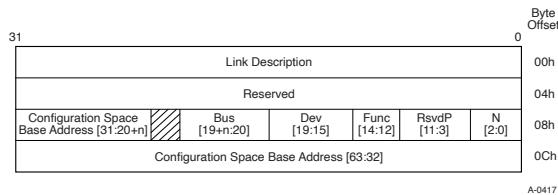


Figure 7-270 Link Address for Link Type 1 §

Table 7-243 Link Address for Link Type 1 §

| Bit Location | Register Description | Attributes |
|-----------------|--|------------|
| 2:0 | N - Encoded number of Bus Number bits | HwInit |
| 14:12 | Function Number | HwInit |
| 19:15 | Device Number | HwInit |
| (19 + n):20 | Bus Number | HwInit |
| 63:($20 + n$) | PCI Express Configuration Space Base Address ($1 \leq n \leq 8$) Note: A Root Complex that does not implement multiple Configuration Spaces is allowed to report this field as 0. | HwInit |

7.9.9 Root Complex Internal Link Control Extended Capability §

The Root Complex Internal Link Control Extended Capability is an optional Capability that controls an internal Root Complex Link between two distinct Root Complex Components. This Capability is valid for RCRBs that declare an Element Type field as Internal Root Complex Link in the Element Self-Description register of the Root Complex Link Declaration Capability structure.

The Root Complex Internal Link Control Extended Capability structure is defined as shown in § Figure 7-271 .

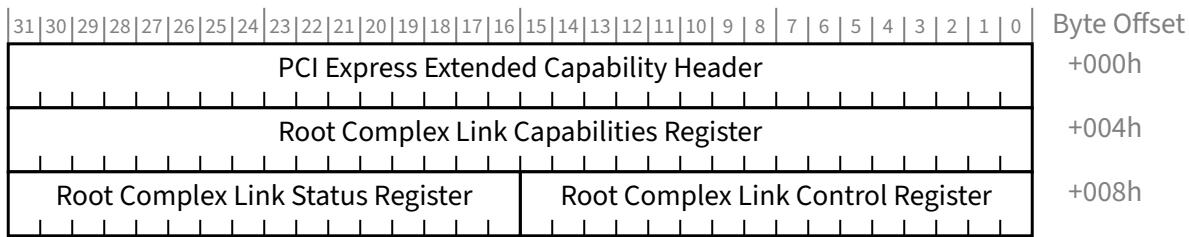


Figure 7-271 Root Complex Internal Link Control Extended Capability §

7.9.9.1 Root Complex Internal Link Control Extended Capability Header (Offset 00h) §

The Extended Capability ID for the Root Complex Internal Link Control Extended Capability is 0006h .

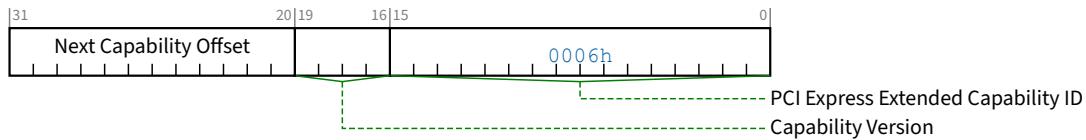


Figure 7-272 Root Complex Internal Link Control Extended Capability Header §

Table 7-244 Root Complex Internal Link Control Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Root Complex Internal Link Control Extended Capability is 0006h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits. | RO |

7.9.9.2 Root Complex Link Capabilities Register (Offset 04h) §

The Root Complex Link Capabilities Register identifies capabilities for this Link.

Base 6.4 vs Base 6.3

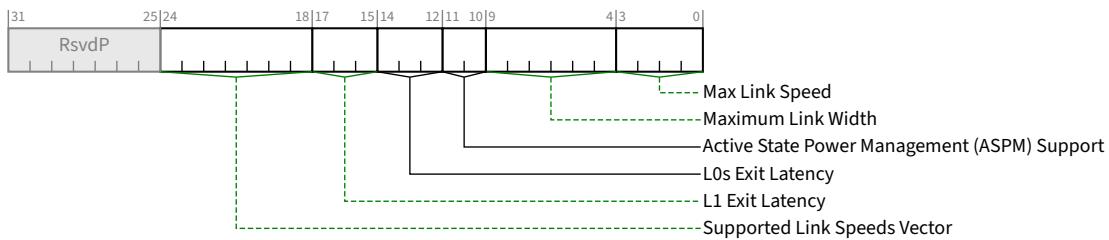


Figure 7-273 Root Complex Link Capabilities Register §

Table 7-245 Root Complex Link Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 3:0 | <p>Max Link Speed - This field indicates the maximum Link speed of the associated Link. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Root Complex Link Capabilities Register) that corresponds to the maximum Link speed. Defined encodings are:</p> <ul style="list-style-type: none"> 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 Others All other encodings are reserved. <p>A Root Complex that does not support this feature must report 0000b in this field.</p> | RO |
| 9:4 | <p>Maximum Link Width - This field indicates the maximum width of the given Link. Defined encodings are:</p> <ul style="list-style-type: none"> 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 01 0000b x16 <p>All other encodings are Reserved. A Root Complex that does not support this feature must report 00 0000b in this field.</p> | RO |
| 11:10 | <p>Active State Power Management (ASPM) Support - This field indicates the level of ASPM supported on the given Link. Defined encodings are:</p> <ul style="list-style-type: none"> 00b No ASPM Support 01b L0s Supported 10b L1 Supported 11b L0s and L1 Supported | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 14:12 | <p>L0s Exit Latency - This field indicates the L0s exit latency for the given Link. The value reported indicates the length of time this Port requires to complete transition from L0s to L0. If L0s is not supported, the value is undefined.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b Less than 64 ns 001b 64 ns to less than 128 ns 010b 128 ns to less than 256 ns 011b 256 ns to less than 512 ns 100b 512 ns to less than 1 μs 101b 1 μs to less than 2 μs 110b 2 μs to 4 μs 111b More than 4 μs | RO |
| 17:15 | <p>L1 Exit Latency - This field indicates the L1 exit latency for the given Link. The value reported indicates the length of time this Port requires to complete transition from ASPM L1 to L0. If ASPM L1 is not supported, the value is undefined.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b Less than 1 μs 001b 1 μs to less than 2 μs 010b 2 μs to less than 4 μs 011b 4 μs to less than 8 μs 100b 8 μs to less than 16 μs 101b 16 μs to less than 32 μs 110b 32 μs to 64 μs 111b More than 64 μs | RO |
| 24:18 | <p>Supported Link Speeds Vector - This field indicates the supported Link speed(s) of the associated Link. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. See § Section 8.2.1 for further requirements.</p> <p>Bit definitions within this field are:</p> <ul style="list-style-type: none"> Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bit 4 32.0 GT/s Bit 5 64.0 GT/s Bit 6 RsvdP | RO |

IMPLEMENTATION NOTE: SUPPORTED LINK SPEEDS WITH EARLIER HARDWARE §

Hardware components compliant to versions prior to the [PCIe-3.0] did not implement the Supported Link Speeds Vector field and instead returned 0000 000b in bits 24:18.

For software to determine the supported Link speeds for components where this field is contains 0000 000b, software can read bits 3:0 of the Root Complex Link Capabilities Register (now defined to be the Max Link Speed field), and interpret the value as follows:

0001b

2.5 GT/s Link speed supported

0010b

5.0 GT/s and 2.5 GT/s Link speeds supported

For such components, the same encoding is also used for the values for the Current Link Speed field (in the Root Complex Link Status Register).

IMPLEMENTATION NOTE: SOFTWARE MANAGEMENT OF LINK SPEEDS WITH FUTURE HARDWARE §

It is strongly encouraged that software primarily utilize the Supported Link Speeds Vector instead of the Max Link Speed field, so that software can determine the exact set of supported speeds on current and future hardware. This can avoid software being confused if a future specification defines Links that do not require support for all slower speeds.

7.9.9.3 Root Complex Link Control Register (Offset 08h) §

The Root Complex Link Control Register controls parameters for this internal Link.

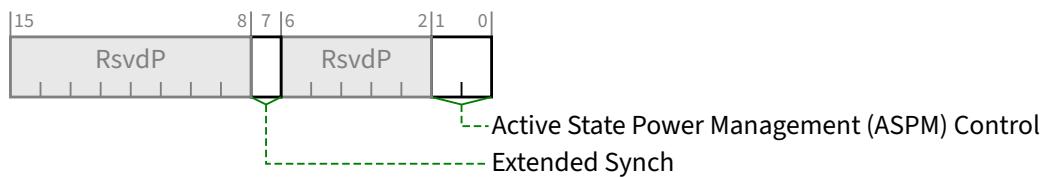


Figure 7-274 Root Complex Link Control Register §

Base 6.4 vs Base 6.3

Table 7-246 Root Complex Link Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 1:0 | <p>Active State Power Management (ASPM) Control - This field controls the level of ASPM enabled on the given Link.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b Disabled 01b L0s Entry Enabled 10b L1 Entry Enabled 11b L0s and L1 Entry Enabled <p>Note: “L0s Entry Enabled” enables the Transmitter to enter L0s . If L0s is supported, the Receiver must be capable of entering L0s even when the Transmitter is disabled from entering L0s (00b or 10b).</p> <p>In Flit Mode, L0s is not supported, bit 0 of this field is ignored and has no effect (i.e., encodings 01b and 00b are equivalent as are encodings 11b and 10b).</p> <p>Default value of this field is implementation specific.</p> <p>Software must not enable L0s in either direction on a given Link unless components on both sides of the Link each support L0s , as indicated by their ASPM Support field values. Otherwise, the result is undefined.</p> <p>ASPM L1 must be enabled by software in the Upstream component on a Link prior to enabling ASPM L1 in the Downstream component on that Link. When disabling ASPM L1, software must disable ASPM L1 in the Downstream component on a Link prior to disabling ASPM L1 in the Upstream component on that Link. ASPM L1 must only be enabled on the Downstream component if both components on a Link support ASPM L1.</p> <p>A Root Complex that does not support this feature for the given internal Link must hardwire this field to 00b.</p> | RW |
| 7 | <p>Extended Synch - This bit when Set forces the transmission of additional Ordered Sets when exiting the L0s state (see § Section 4.2.5.6) and when in the Recovery state (see § Section 4.2.7.4.1). This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and Symbol lock before the Link enters the L0 state and resumes communication.</p> <p>A Root Complex that does not support this feature for the given internal Link must hardwire this bit to 0b.</p> <p>In Flit Mode, this bit is ignored and has no effect since L0s is not supported.</p> <p>Default value for this bit is 0b.</p> | RW |

7.9.9.4 Root Complex Link Status Register (Offset 0Ah) §

The Root Complex Link Status Register provides information about Link specific parameters.

Base 6.4 vs Base 6.3

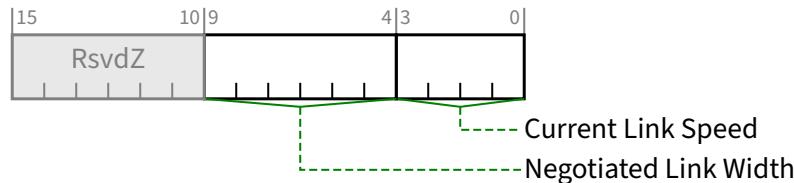


Figure 7-275 Root Complex Link Status Register §

Table 7-247 Root Complex Link Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | <p>Current Link Speed - This field indicates the negotiated Link speed of the given Link. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Root Complex Link Capabilities Register) that corresponds to the current Link speed.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 <p>All other encodings are Reserved.</p> <p>The value in this field is undefined when the Link is not up. A Root Complex that does not support this feature must report 0000b in this field.</p> | RO |
| 9:4 | <p>Negotiated Link Width - This field indicates the negotiated width of the given Link. This includes the Link Width determined during initial link training as well changes that occur after initial link training (e.g., L0p)</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 01 0000b x16 <p>All other encodings are Reserved. The value in this field is undefined when the Link is not up. A Root Complex that does not support this feature must hardwire this field to 00 0000b.</p> | RO |

7.9.10 Root Complex Event Collector Endpoint Association Extended Capability §

The Root Complex Event Collector Endpoint Association Extended Capability is implemented by Root Complex Event Collectors . It declares the RCIEPs supported by the Root Complex Event Collector . A Root Complex Event Collector must

implement the Root Complex Event Collector Endpoint Association Extended Capability ; no other PCI Express Device Function is permitted to implement this Capability.

The Root Complex Event Collector Endpoint Association Extended Capability , as shown in § Figure 7-276 , consists of the PCI Express Extended Capability header followed by a DWORD bitmap enumerating RCiEPs on the same Bus, and optionally an additional range of Bus Numbers that may contain RCiEPs associated with the Root Complex Event Collector . Functions other than RCiEPs (e.g., Root Ports) contained in the range described by this Capability are not associated with this Root Complex Event Collector.

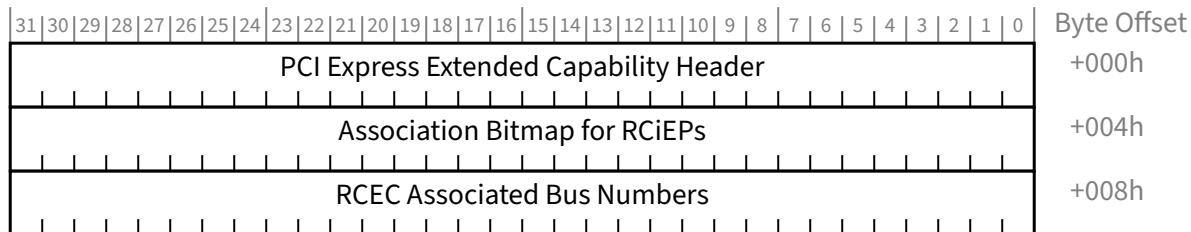


Figure 7-276 Root Complex Event Collector Endpoint Association Extended Capability §

7.9.10.1 Root Complex Event Collector Endpoint Association Extended Capability Header (Offset 00h) §

The Extended Capability ID for the Root Complex Event Collector Endpoint Association Extended Capability is 0007h . § Figure 7-277 details allocation of fields in the Root Complex Event Collector Endpoint Association Extended Capability Header ; § Table 7-248 provides the respective bit definitions.

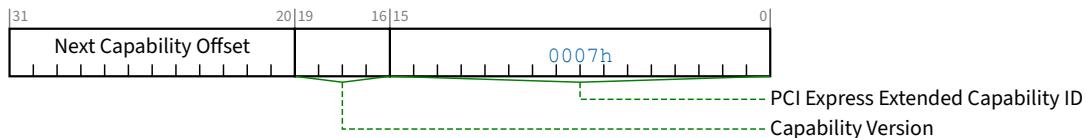


Figure 7-277 Root Complex Event Collector Endpoint Association Extended Capability Header §

Table 7-248 Root Complex Event Collector Endpoint Association Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | <p>PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.</p> <p>The Extended Capability ID for the Root Complex Event Collector Endpoint Association Extended Capability is 0007h .</p> | RO |
| 19:16 | <p>Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.</p> <p>Must be 2h if the Extended Capability contains the RCEC Associated Bus Numbers Register (see § Section 7.9.10.3). Must be 1h otherwise.</p> | RO |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:20 | <p>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.</p> <p>For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.</p> <p>The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.</p> | RO |

7.9.10.2 Association Bitmap for RCiEPs (Offset 04h) §

The Association Bitmap for RCiEPs is a read-only register that sets the bits corresponding to the Device Numbers of RCiEPs associated with the Root Complex Event Collector on the same Bus Number as the Event Collector itself. The bit corresponding to the Device Number of the Root Complex Event Collector must always be Set.

7.9.10.3 RCEC Associated Bus Numbers Register (Offset 08h) §

The RCEC Associated Bus Numbers Register is a read-only register that indicates an additional range of Bus Numbers containing RCiEPs associated with this Root Complex Event Collector . It is permitted for Functions other than RCiEPs , including Root Ports, to appear within the Association Bus Range. Only RCiEPs in the range are associated with this Root Complex Event Collector . This register is present if the Capability Version is 2h or greater.

This register does not indicate association between an Event Collector and any Virtual Functions within the Association Bus Range (see § Section 9.2.1.2). This register does not indicate association between an Event Collector and any Function on the same Bus Number as the Event Collector itself, however it is permitted for the Association Bus Range to include the Bus Number of the Root Complex Event Collector.

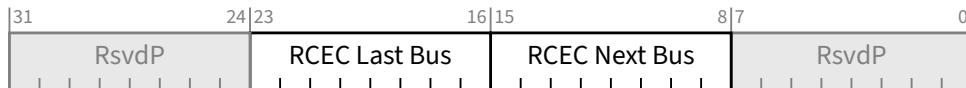


Figure 7-278 RCEC Associated Bus Numbers Register §

Table 7-249 RCEC Associated Bus Numbers Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:8 | RCEC Next Bus - This field contains the lowest additional bus number containing RCiEPs associated with this Root Complex Event Collector. If all of the Devices associated with this Root Complex Event Collector are on the same bus as the Event Collector, then this field must be set to FFh. | HwInit |
| 23:16 | <p>RCEC Last Bus - This field contains the highest additional bus number containing RCiEPs associated with this Root Complex Event Collector.</p> <p>If all of the Devices associated with this Root Complex Event Collector are on the same bus as the Event Collector, then this field must be set to 00h.</p> | HwInit |

IMPLEMENTATION NOTE: RCEC ASSOCIATED BUS NUMBER COMPATIBILITY WITH LEGACY SOFTWARE §

Legacy software may not support the use of the RCEC Associated Bus Numbers Register as a mechanism to associate Devices with a RCEC. Such software may see events in the RCEC from Devices on different bus numbers that it does not consider to be associated with the Root Complex Event Collector. System Software is strongly encouraged to report all events seen on the Root Complex Event Collector, regardless of whether or not it can determine association.

7.9.11 Multicast Extended Capability §

Multicast is an optional normative functionality that is controlled by the Multicast Extended Capability structure. The Multicast Extended Capability is applicable to Root Ports, RCRBs, Switch Ports, Endpoint Functions, and RCiEPs. It is not applicable to PCI Express to PCI/PCI-X Bridges.

Multicast support is optional in SR-IOV devices. If a VF implements a Multicast capability, its associated PF must implement a Multicast capability.

Multicast support is optional in SIOV Devices. If an SDI requires multicast functionality, its associated PF must implement the Multicast Extended capability. SDIs must use the multicast settings from the Multicast Extended capability of their associated PF.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

In the cases of a Switch or Root Complex or a component that contains multiple Functions, multiple copies of this Capability structure are required - one for each Endpoint Function, Switch Port, or Root Port that supports Multicast. To provide implementation efficiencies, certain fields within each of the Multicast Extended Capability structures within a component must be programmed the same and results are indeterminate if this is not the case. The fields and registers that must be configured with the same values include MC_Enable, MC_Num_Group, MC_Base_Address and MC_Index_Position. These same fields in an Endpoint's Multicast Extended Capability structure must match those configured into a Multicast Extended Capability structure of the Switch or Root Complex above the Endpoint or in which the RCiEP is integrated.

Base 6.4 vs Base 6.3

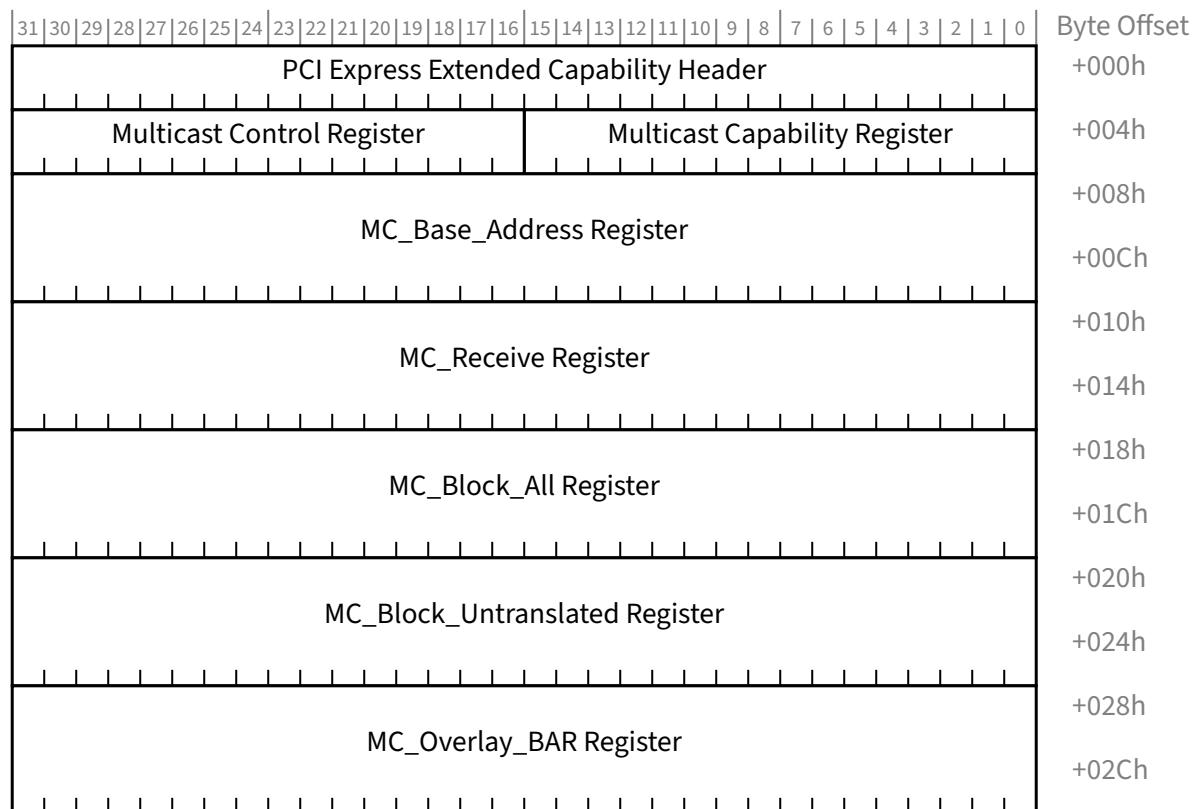


Figure 7-279 Multicast Extended Capability Structure §

7.9.11.1 Multicast Extended Capability Header (Offset 00h) §

§ Figure 7-280 details allocation of the fields in the Multicast Extended Capability Header and § Table 7-250 provides the respective bit definitions.

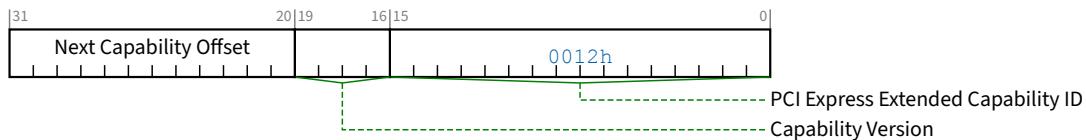


Figure 7-280 Multicast Extended Capability Header §

Table 7-250 Multicast Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | PCI Express Extended Capability ID for the <u>Multicast Extended Capability</u> is 0012h . | |
| 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. | <u>RO</u> |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | <u>RO</u> |

7.9.11.2 Multicast Capability Register (Offset 04h) §

§ Figure 7-281 details allocation of the fields in the Multicast Capability Register and § Table 7-251 provides the respective bit definitions.

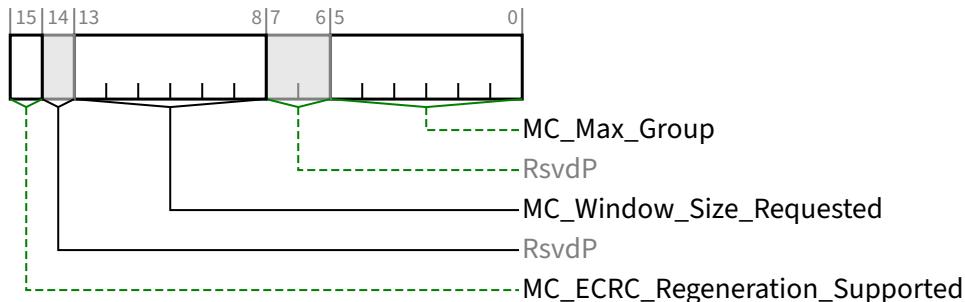


Figure 7-281 Multicast Capability Register §

Table 7-251 Multicast Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------------|
| 5:0 | MC_Max_Group - Value indicates the maximum number of Multicast Groups that the component supports, encoded as M-1. A value of 00h indicates that one Multicast Group is supported. For VFs, this field is <u>RsvdP</u> . The value from the associated PF applies. | <u>RO</u> <u>VF RsvdP</u> |
| 13:8 | MC_Window_Size_Requested - In Endpoints, the log ₂ of the Multicast Window size requested. <u>RsvdP</u> in Switch and Root Ports. For VFs, this field is <u>RsvdP</u> . The value from the associated PF applies. | <u>RO</u> <u>VF RsvdP</u> |
| 15 | MC_ECRC_Regeneration_Supported - If Set, indicates that ECRC regeneration is supported. This bit must not be Set unless the Function supports Advanced Error Reporting, and the <u>ECRC Check Capable</u> bit in the <u>Advanced Error Capabilities and Control Register</u> is also Set. However, if ECRC regeneration is supported, its operation is not contingent upon the setting of the <u>ECRC Check Enable</u> bit in the <u>Advanced Error Capabilities and Control Register</u> . This bit is applicable to Switch and Root Ports and is <u>RsvdP</u> in all other Functions. | <u>RO / RsvdP</u> |

7.9.11.3 Multicast Control Register (Offset 06h) §

§ Table 7-252 details allocation of the fields in the Multicast Control Register and § Table 7-252 provides the respective bit definitions.

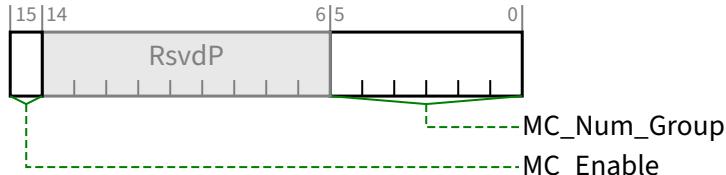


Figure 7-282 Multicast Control Register §

Table 7-252 Multicast Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------------------------|
| 5:0 | <p>MC_Num_Group - Value indicates the number of Multicast Groups configured for use, encoded as N-1. The default value of 00 0000b indicates that one Multicast Group is configured for use. Behavior is undefined if value exceeds MC_Max_Group . This parameter indirectly defines the upper limit of the Multicast address range. This field is ignored if MC_Enable is Clear. Default value is 00 0000b.</p> <p>For VFs, this field is <u>RsvdP</u> . The value from the associated PF applies.</p> | <u>RW</u> <u>VF RsvdP</u> |
| 15 | MC_Enable - When Set, the Multicast mechanism is enabled for the component. Default value is 0b. | <u>RW</u> |

7.9.11.4 MC_Base_Address Register (Offset 08h) §

The MC_Base_Address Register contains the MC_Base_Address and the MC_Index_Position . § Figure 7-283 details allocation of the fields in the MC_Base_Address Register and § Table 7-253 provides the respective bit definitions.

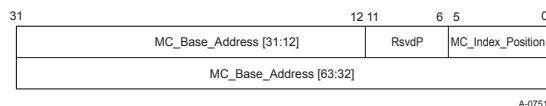


Figure 7-283 MC_Base_Address Register §

Table 7-253 MC_Base_Address Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------------------------|
| 5:0 | <p>MC_Index_Position - The location of the LSB of the Multicast Group number within the address. Behavior is undefined if this value is less than 12 and MC_Enable is Set. Default is 0.</p> <p>For VFs, this field is <u>RsvdP</u> . The value from the associated PF applies.</p> | <u>RW</u> <u>VF RsvdP</u> |

| Bit Location | Register Description | Attributes |
|--------------|---|------------------------------|
| 63:12 | <p>MC_Base_Address - The base address of the Multicast address range. The behavior is undefined if MC_Enable is Set and bits in this field corresponding to address bits that contain the Multicast Group number or address bits less than MC_Index_Position are non-zero. Default is 0.</p> <p>For VFs, this field is <u>RsvdP</u>. The value from the associated PF applies.</p> | <u>RW</u> <u>VF RsvdP</u> |

7.9.11.5 MC_Receive Register (Offset 10h) §

The MC_Receive Register provides a bit vector denoting which Multicast groups the Function should accept, or in the case of Switch and Root Complex Ports, forward Multicast TLPs. This register is required in all Functions that implement the MC Capability structure.

§ Figure 7-284 details allocation of the fields in the MC_Receive Register and § Table 7-254 provides the respective bit definitions.

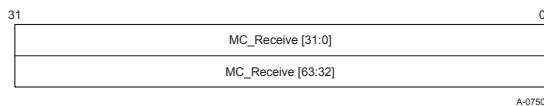


Figure 7-284 MC_Receive Register §

Table 7-254 MC_Receive Register §

| Bit Location | Register Description | Attributes |
|-----------------|--|--------------|
| MC_Max_Group :0 | MC_Receive - For each bit that's Set, this Function gets a copy of any Multicast TLPs for the associated Multicast Group. Bits above MC_Num_Group are ignored by hardware. Default value of each bit is 0b. | <u>RW</u> |
| All other bits | Reserved | <u>RsvdP</u> |

7.9.11.6 MC_Block_All Register (Offset 18h) §

The MC_Block_All Register provides a bit vector denoting which Multicast groups the Function should block. This register is required in all Functions that implement the MC Capability structure.

§ Figure 7-285 details allocation of the fields in the MC_Block_All Register and § Table 7-255 provides the respective bit definitions.

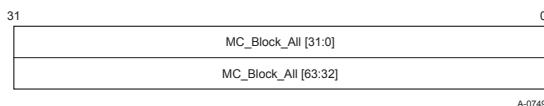


Figure 7-285 MC_Block_All Register §

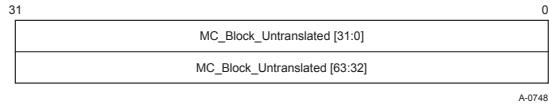
Table 7-255 MC_Block_All Register

| Bit Location | Register Description | Attributes |
|-----------------|---|------------|
| MC_Max_Group :0 | MC_Block_All - For each bit that is Set, this Function is blocked from sending TLPs to the associated Multicast Group. Bits above MC_Num_Group are ignored by hardware. Default value of each bit is 0b. | RW |
| All other bits | Reserved | RsvdP |

7.9.11.7 MC_Block_Untranslated Register (Offset 20h)

The MC_Block_Untranslated Register is used to determine whether or not a TLP that includes an Untranslated Address should be blocked. This register is required in all Functions that implement the MC Capability structure. However, an Endpoint Function that does not implement the ATS capability may implement this register as RsvdP.

Figure 7-286 details allocation of the fields in the MC_Block_Untranslated Register and Table 7-256 provides the respective bit definitions.

*Figure 7-286 MC_Block_Untranslated Register**Table 7-256 MC_Block_Untranslated Register*

| Bit Location | Register Description | Attributes |
|-----------------|--|------------|
| MC_Max_Group :0 | MC_Block_Untranslated - For each bit that is Set, this Function is blocked from sending TLPs containing Untranslated Addresses to the associated MCG. Bits above MC_Num_Group are ignored by hardware. Default value of each bit is 0b. | RW |
| All other bits | Reserved | RsvdP |

7.9.11.8 MC_Overlay_BAR Register (Offset 28h)

The MC_Overlay_BAR Register is required in Switch and Root Complex Ports that support the Multicast Extended Capability and not implemented in Endpoints. Software must interpret the Device/Port Type field in the PCI Express Capabilities Register to determine if the MC_Overlay_BAR Register is present in a Function.

The MC_Overlay_BAR specifies the base address of a window in unicast space onto which Multicast TLPs going out an Egress Port are overlaid by a process of address replacement. This allows a single BAR in an Endpoint attached to the Switch or Root Port to be used for both unicast and Multicast traffic. At a Switch Upstream Port, it allows the Multicast address range, or a portion of it, to be overlaid onto host memory.

Figure 7-287 details allocation of the fields in the MC_Overlay_BAR Register and Table 7-257 provides the respective bit definitions.

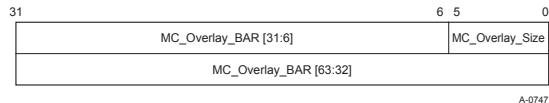


Figure 7-287 MC_Overlay_BAR Register §

Table 7-257 MC_Overlay_BAR Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 5:0 | MC_Overlay_Size - If 6 or greater, specifies the size in bytes of the overlay aperture as a power of 2. If less than 6, disables the overlay mechanism. Default value is 00 0000b. | RW |
| 63:6 | MC_Overlay_BAR - Specifies the base address of the window onto which MC TLPs passing through this Function will be overlaid. Default value is 0. | RW |

7.9.12 Dynamic Power Allocation Extended Capability (DPA Capability) §

The DPA Capability structure is shown in § Figure 7-288 .

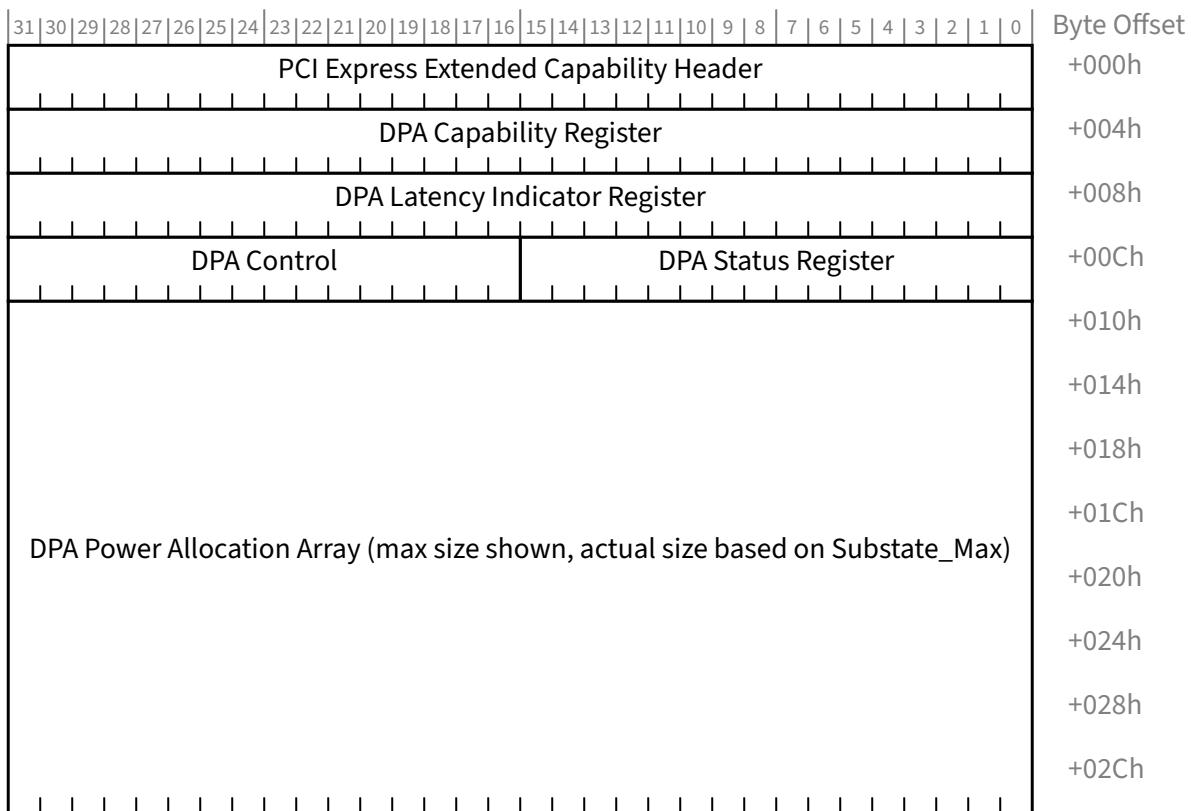


Figure 7-288 Dynamic Power Allocation Extended Capability Structure §

7.9.12.1 DPA Extended Capability Header (Offset 00h) §

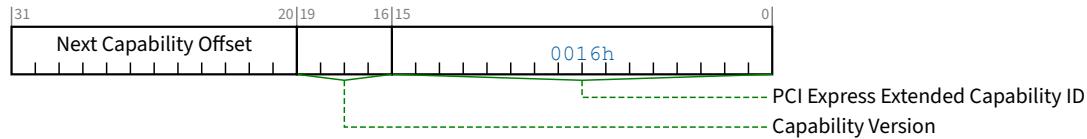


Figure 7-289 DPA Extended Capability Header §

Table 7-258 DPA Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the DPA Extended Capability is 0016h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

7.9.12.2 DPA Capability Register (Offset 04h) §

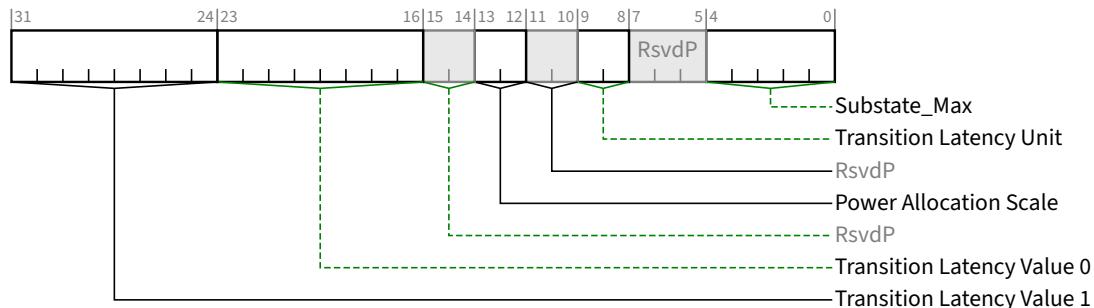


Figure 7-290 DPA Capability Register §

Table 7-259 DPA Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 4:0 | Substate_Max - Value indicates the maximum substate number, which is the total number of supported substates minus one. A value of 0 0000b indicates support for one substate. | RO |

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|---|------------|-------|------------|-------|------------|--------|------------|----------|----|
| 9:8 | <p>Transition Latency Unit (Tlunit) - A substate's Transition Latency Value is multiplied by the <u>Transition Latency Unit</u> to determine the maximum Transition Latency for the substate.</p> <p>Defined encodings are</p> <table> <tr> <td>00b</td> <td>1 ms</td> </tr> <tr> <td>01b</td> <td>10 ms</td> </tr> <tr> <td>10b</td> <td>100 ms</td> </tr> <tr> <td>11b</td> <td>Reserved</td> </tr> </table> | 00b | 1 ms | 01b | 10 ms | 10b | 100 ms | 11b | Reserved | RO |
| 00b | 1 ms | | | | | | | | | |
| 01b | 10 ms | | | | | | | | | |
| 10b | 100 ms | | | | | | | | | |
| 11b | Reserved | | | | | | | | | |
| 13:12 | <p>Power Allocation Scale (PAS) - The encodings provide the scale to determine power allocation per substate in Watts. The value corresponding to the substate in the <u>Substate Power Allocation</u> field is multiplied by this field to determine the power allocation for the substate.</p> <p>Defined encodings are</p> <table> <tr> <td>00b</td> <td>10.0x</td> </tr> <tr> <td>01b</td> <td>1.0x</td> </tr> <tr> <td>10b</td> <td>0.1x</td> </tr> <tr> <td>11b</td> <td>0.01x</td> </tr> </table> | 00b | 10.0x | 01b | 1.0x | 10b | 0.1x | 11b | 0.01x | RO |
| 00b | 10.0x | | | | | | | | | |
| 01b | 1.0x | | | | | | | | | |
| 10b | 0.1x | | | | | | | | | |
| 11b | 0.01x | | | | | | | | | |
| 23:16 | <p>Transition Latency Value 0 (XlcY0) - This value is multiplied by the <u>Transition Latency Unit</u> to determine the maximum Transition Latency for the substate</p> | RO | | | | | | | | |
| 31:24 | <p>Transition Latency Value 1 (XlcY1) - This value is multiplied by the <u>Transition Latency Unit</u> to determine the maximum Transition Latency for the substate.</p> | RO | | | | | | | | |

7.9.12.3 DPA Latency Indicator Register (Offset 08h) §

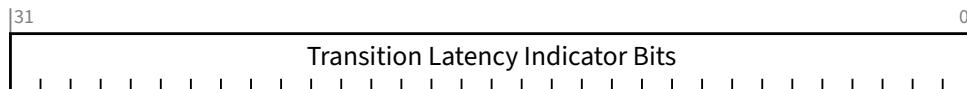


Figure 7-291 DPA Latency Indicator Register §

Table 7-260 DPA Latency Indicator Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:0 | <p>Transition Latency Indicator Bits - Each bit indicates which Transition Latency Value is associated with the corresponding substate. A value of 0b indicates <u>Transition Latency Value 0</u>; a value of 1b indicates <u>Transition Latency Value 1</u>.</p> <p>Only bits [Substate_Max :0] are defined. Bits above Substate_Max are RsvdP .</p> | RO |

7.9.12.4 DPA Status Register (Offset 0Ch) §

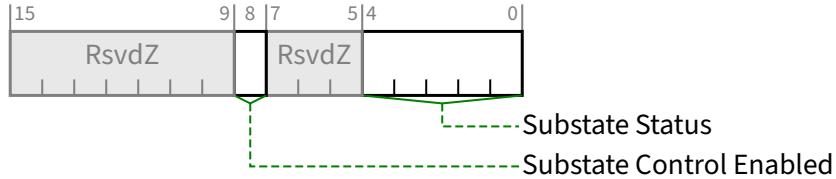


Figure 7-292 DPA Status Register §

Table 7-261 DPA Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 4:0 | <p>Substate Status - Indicates current substate for this Function. Default is 0 0000b.</p> | RO |
| 8 | <p>Substate Control Enabled - Used by software to disable the Substate Control field in the DPA Control Register. Hardware sets this bit following a Conventional Reset or FLR. Software clears this bit by writing a 1b to it. Software is unable to set this bit directly.</p> <p>When this bit is Set, the <u>Substate Control</u> field determines the current substate.</p> <p>When this bit is Clear, the <u>Substate Control</u> field has no effect on the current substate.</p> <p>Default value is 1b.</p> | RW1C |

7.9.12.5 DPA Control Register (Offset 0Eh) §

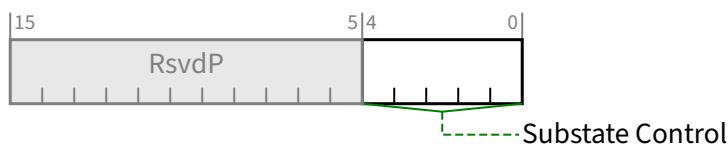


Figure 7-293 DPA Control Register §

Table 7-262 DPA Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 4:0 | <p>Substate Control - Used by software to configure the Function substate. Software writes the substate value in this field to initiate a substate transition.</p> <p>When the <u>Substate Control Enabled</u> bit in the DPA Status Register is Set, this field determines the Function substate.</p> <p>When the <u>Substate Control Enabled</u> bit in the DPA Status Register is Clear, this field has no effect on the Function substate.</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|---------------------------|------------|
| | Default value is 0 0000b. | |

7.9.12.6 DPA Power Allocation Array §

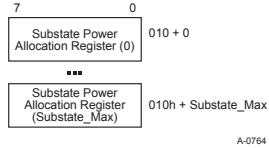


Figure 7-294 DPA Power Allocation Array §

Each Substate Power Allocation register indicates the power allocation value for its associated substate. The number of Substate Power Allocation registers implemented must be equal to the number of substates supported by Function, which is Substate_Max plus one.

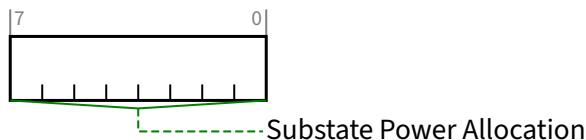


Figure 7-295 Substate Power Allocation Register (0 to Substate_Max) §

Table 7-263 Substate Power Allocation Register (0 to Substate_Max) §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | Substate Power Allocation - The value in this field is multiplied by the Power Allocation Scale to determine power allocation in Watts for the associated substate. | RO |

7.9.13 TPH Requester Extended Capability §

The TPH Requester Extended Capability structure is required for all Functions that are capable of generating Request TLPs with TPH. For a Multi-Function Device, this capability must be present in each Function that is capable of generating Request TLPs with TPH.

The capability is optional for PFs and VFs. However, if a VF associated with a given PF contains the capability, all VFs associated with that PF must contain the capability.

For fields in the TPH Requester Capability Register (offset 04h), all VFs associated with a given PF must have the same values in all fields, but the PF's fields may have values different from those in its VFs.

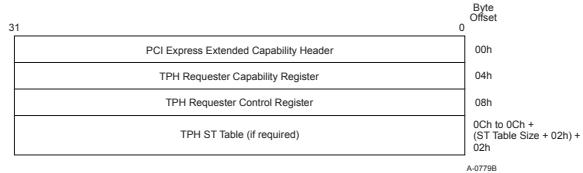


Figure 7-296 TPH Extended Capability Structure §

7.9.13.1 TPH Requester Extended Capability Header (Offset 00h) §

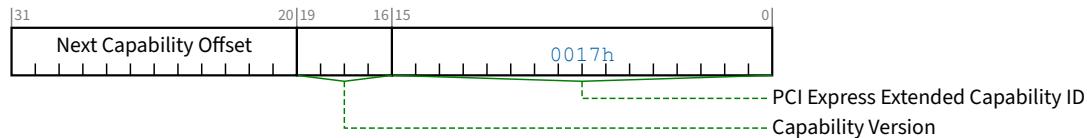


Figure 7-297 TPH Requester Extended Capability Header §

Table 7-264 TPH Requester Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | <p>PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.</p> <p>PCI Express Extended Capability ID for the TPH Requester Extended Capability is 0017h .</p> | RO |
| 19:16 | <p>Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.</p> <p>Must be 1h for this version of the specification.</p> | RO |
| 31:20 | <p>Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.</p> | RO |

7.9.13.2 TPH Requester Capability Register (Offset 04h)

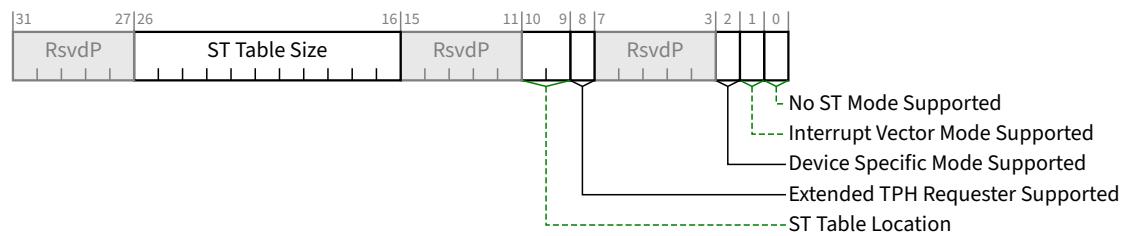


Figure 7-298 TPH Requester Capability Register

Table 7-265 TPH Requester Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | No ST Mode Supported - If set indicates that the Function supports the <u>No ST Mode</u> of operation. This mode is required to be supported by all Functions that implement this Capability structure. This bit must have a value of 1b. | RO |
| 1 | Interrupt Vector Mode Supported - If set indicates that the Function supports the Interrupt Vector Mode of operation. | RO |
| 2 | Device Specific Mode Supported - If set indicates that the Function supports the Device Specific Mode of operation. | RO |
| 8 | Extended TPH Requester Supported - If Set indicates that the Function is capable of generating Requests with additional TPH information using the <u>TPH TLP Prefix</u> . See § Section 2.2.7.1.1 for additional details. | RO |
| 10:9 | ST Table Location - Value indicates if and where the ST Table is located. Defined Encodings are: 00b ST Table is not present 01b ST Table is located in the TPH Requester Extended Capability structure 10b ST Table is located in the MSI-X Table (see § Section 7.7.2) 11b Reserved A Function that only supports the <u>No ST Mode</u> of operation must have a value of 00b in this field. A Function may report a value of 10b only if it implements an MSI-X Capability. | RO |
| 26:16 | ST Table Size - Value indicates the maximum number of ST Table entries the Function may use. Software reads this field to determine the <u>ST Table Size N</u> , which is encoded as N-1. For example, a returned value of 000 0000 0011b indicates a table size of four entries. There is an upper limit of 64 entries when the ST Table is located in the <u>TPH Requester Extended Capability structure</u> . When the ST Table is located in the MSI-X Table, this value is limited by the size of the MSI-X Table. This field is only applicable for Functions that implement an ST Table as indicated by the <u>ST Table Location</u> field. Otherwise, the value in this field is undefined. | RO |

7.9.13.3 TPH Requester Control Register (Offset 08h) §

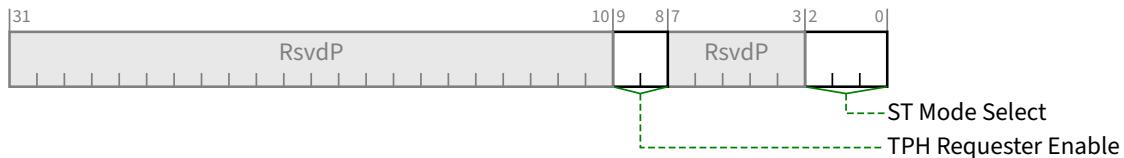
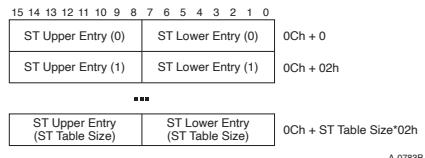


Figure 7-299 TPH Requester Control Register §

Table 7-266 TPH Requester Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>ST Mode Select - selects the ST Mode of operation.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 000b No ST Mode 001b Interrupt Vector Mode 010b Device Specific Mode others reserved for future use <p>Functions that support only the <u>No ST Mode</u> of operation must hardwire this field to 000b.</p> <p>Function operation is undefined if software enables a mode of operation that does not correspond to a mode supported by the Function.</p> <p>The default value of this field is 000b.</p> <p>See § Section 6.17.3 for details on ST modes of operation.</p> | RW |
| 9:8 | <p>TPH Requester Enable - Controls the ability to issue Request TLPs using either TPH or Extended TPH.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00b Function operating as a Requester is not permitted to issue Requests with TPH or Extended TPH 01b Function operating as a Requester is permitted to issue Requests with TPH and is not permitted to issue Requests with Extended TPH 10b Reserved 11b Function operating as a Requester is permitted to issue Requests with TPH and Extended TPH <p>Functions that advertise that they do not support Extended TPH are permitted to hardwire bit 9 of this field to 0b.</p> <p>The default value of this field is 00b.</p> | RW |

7.9.13.4 TPH ST Table (Starting from Offset 0Ch) §*Figure 7-300 TPH ST Table §*

The TPH ST Table must be implemented in the TPH Requester Extended Capability structure if the value of the ST Table Location field is 01b. For all other values, the ST Entry registers must not be implemented. Each implemented ST Entry is 16 bits. The number of ST Entry registers implemented must be equal to the number of ST Table entries supported by the Function, which is the value of the ST Table Size field plus one.



Figure 7-301 TPH ST Table Entry §

Table 7-267 TPH ST Table Entry §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | ST Lower - This field contains the lower 8 bits of a Steering Tag. Default value of this field is 00h. | RW |
| 15:8 | ST Upper - If the Function's Extended TPH Requester Supported bit is Set, then this field contains the upper 8 bits of a Steering Tag. Otherwise, this field is RsvdP. Default value of this field is 00h. | RW |

7.9.14 DPC Extended Capability §

The Downstream Port Containment (DPC) Extended Capability is an optional normative capability that provides a mechanism for Downstream Ports to contain uncorrectable errors and enable software to recover from them. See § Section 6.2.11 . This capability may be implemented by a Root Port or a Switch Downstream Port. It is not applicable to any other Device/Port type.

If a Downstream Port implements the DPC Extended Capability, that Port must also be capable of reporting the DL_Active state, and indicate so by Setting the Data Link Layer Link Active Reporting Capable bit in the Link Capabilities Register . See § Section 7.5.3.6 .

If a Downstream Port implements the DPC Extended Capability, it is strongly recommended for that Port to support ERR_COR Subclass capability, and indicate so by Setting the ERR_COR Subclass Capable bit in the Device Capabilities Register . See § Section 7.5.3.3 .

The various RP PIO registers must be implemented only by Root Ports that support RP Extensions for DPC , as indicated by the RP Extensions for DPC bit in the DPC Capability Register .

Base 6.4 vs Base 6.3

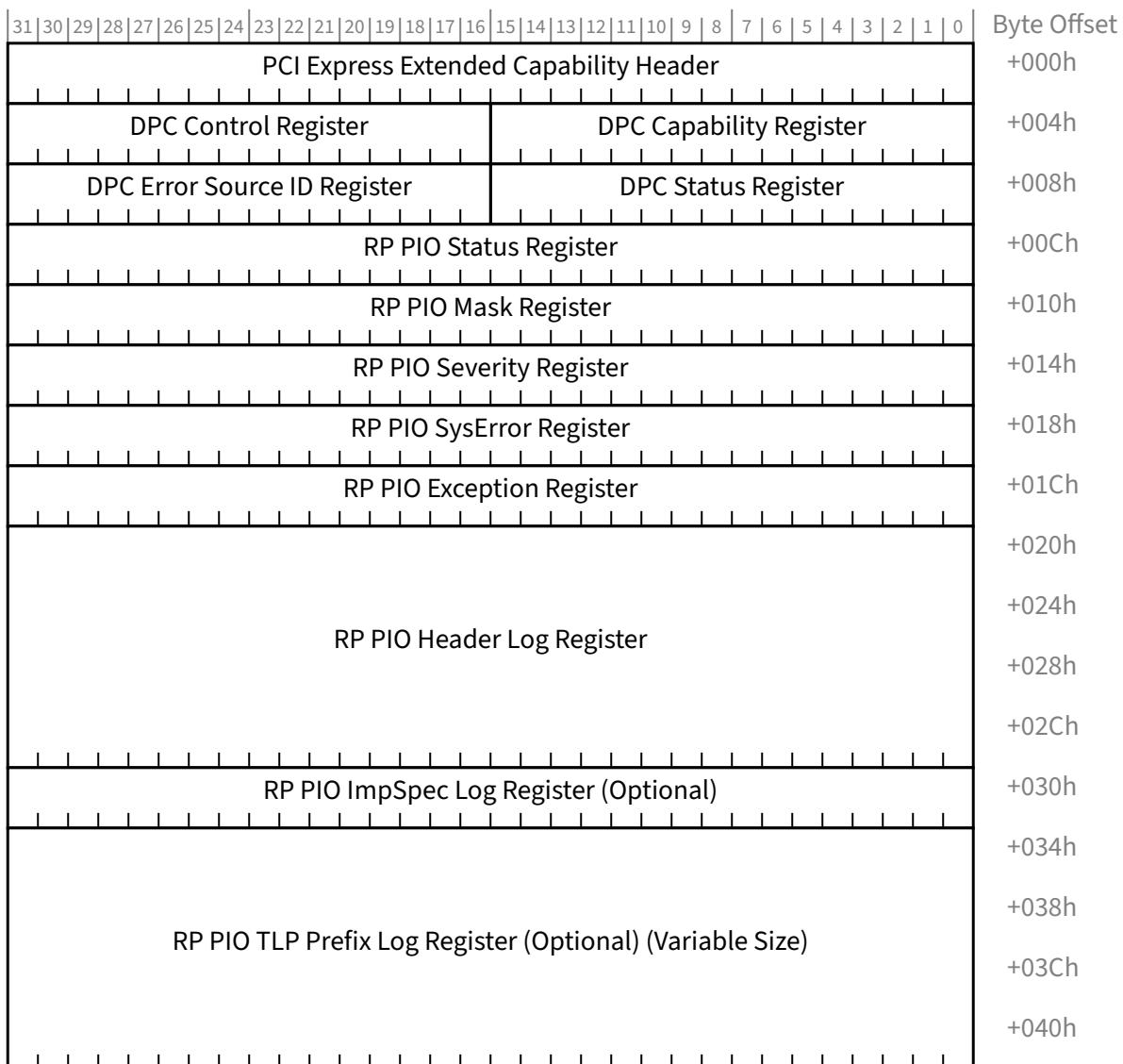


Figure 7-302 DPC Extended Capability – Non-Flit Mode §

Base 6.4 vs Base 6.3

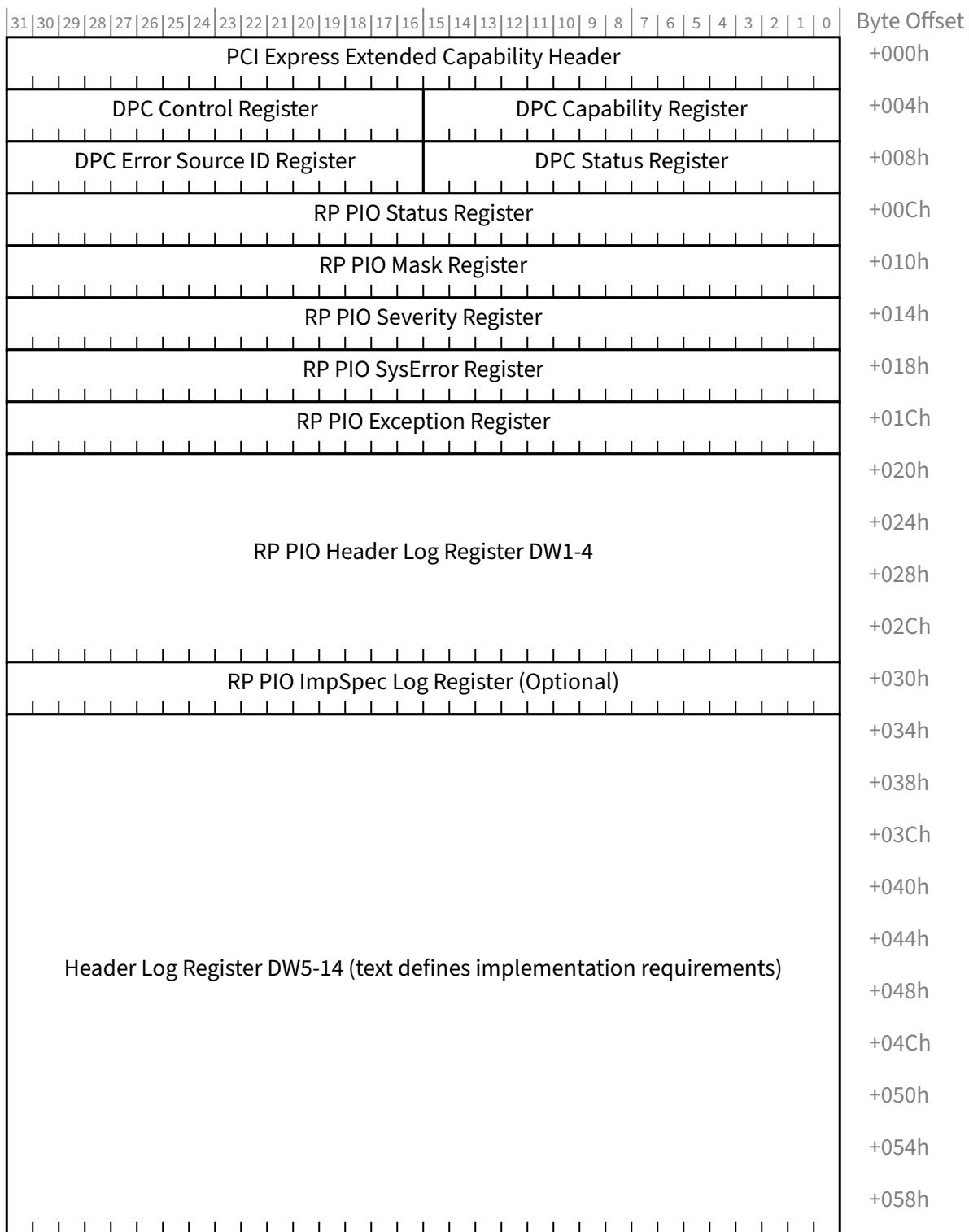


Figure 7-303 DPC Extended Capability – Flit Mode §

7.9.14.1 DPC Extended Capability Header (Offset 00h) §

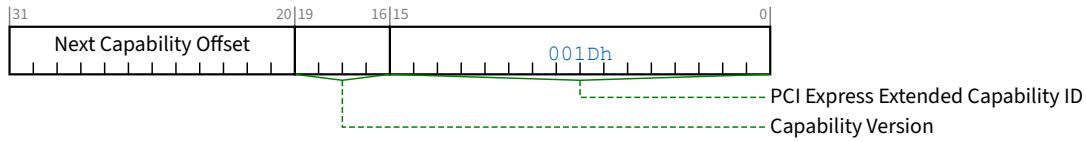


Figure 7-304 DPC Extended Capability Header §

Table 7-268 DPC Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the DPC Extended Capability is 001Dh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

7.9.14.2 DPC Capability Register (Offset 04h) §

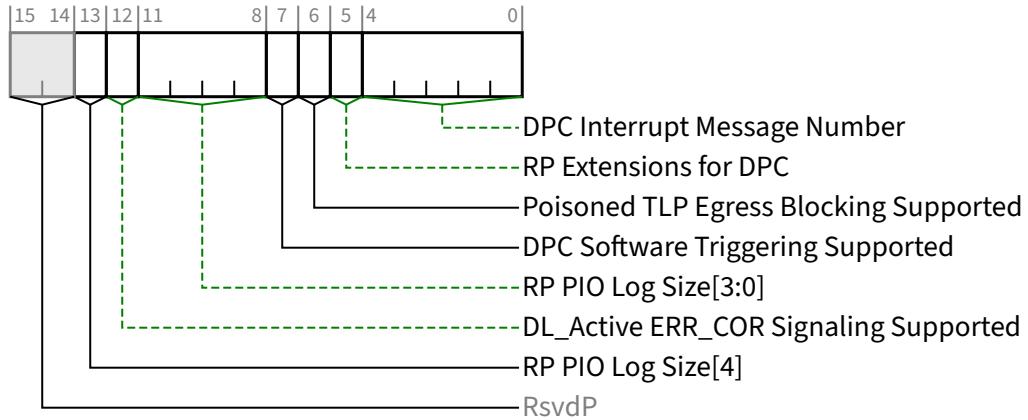


Figure 7-305 DPC Capability Register §

Table 7-269 DPC Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 4:0 | <p>DPC Interrupt Message Number - When MSI/MSI-X is implemented, this field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with the DPC Capability structure.</p> <p>For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the <u>Multiple Message Enable</u> field in the <u>Message Control Register for MSI</u>.</p> <p>For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.</p> | RO |
| 5 | RP Extensions for DPC - If Set, this bit indicates that a Root Port supports a defined set of DPC Extensions that are specific to Root Ports. Switch Downstream Ports must not Set this bit. | RO |
| 6 | Poisoned TLP Egress Blocking Supported - If Set, this bit indicates that the Root Port or Switch Downstream Port supports the ability to block the transmission of a poisoned TLP from its Egress Port. Root Ports that support RP Extensions for DPC must Set this bit. | RO |
| 7 | DPC Software Triggering Supported - If Set, this bit indicates that a Root Port or Switch Downstream Port supports the ability for software to trigger DPC. Root Ports that support RP Extensions for DPC must Set this bit. | RO |
| 11:8 | <p>RP PIO Log Size[3:0] - This field indicates how many DWORDs are allocated for the RP PIO log registers, comprised by the RP PIO Header Log, the RP PIO ImpSpec Log, and RP PIO TLP Prefix Log.</p> <ul style="list-style-type: none"> • If the Root Port does not support <u>RP Extensions for DPC</u>, the value of this field must be <u>Zero</u>. • If the Root Port supports <u>RP Extensions for DPC</u> but does not support Flit Mode, the value of this field must be 4 or greater. • If the Root Port supports both <u>RP Extensions for DPC</u> and Flit Mode, see <u>§ Section 6.2.11.3</u> for requirements. <p>See <u>§ Section 7.9.14.11</u>, <u>§ Section 7.9.14.12</u>, and <u>§ Section 7.9.14.13</u>.</p> | RO |
| 12 | DL_Active ERR_COR Signaling Supported - If Set, this bit indicates that the Root Port or Switch Downstream Port supports the ability to signal with ERR_COR when the Link transitions to the <u>DL_Active</u> state. Root Ports that support RP Extensions for DPC must Set this bit. | RO |
| 13 | RP PIO Log Size[4] - This bit is an extension of <u>RP PIO Log Size[3:0]</u> for use in Flit Mode. If Flit Mode is not supported, this bit is <u>RsvdP</u> . | RO / RsvdP |

7.9.14.3 DPC Control Register (Offset 06h) §

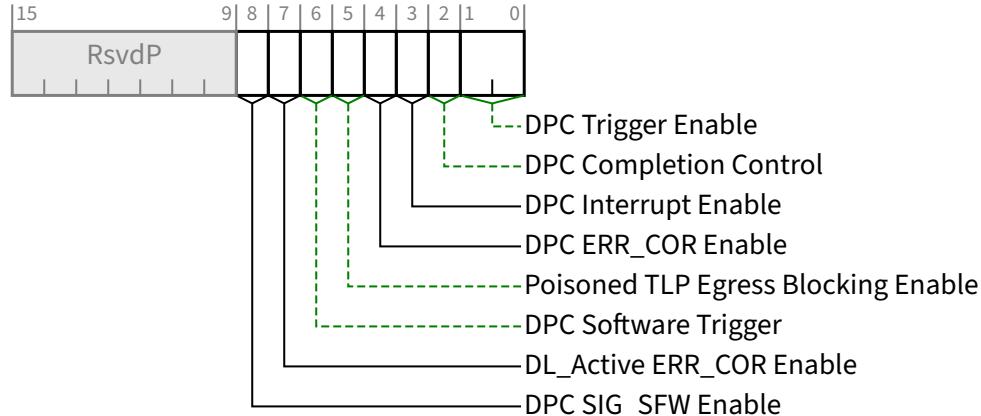


Figure 7-306 DPC Control Register §

Table 7-270 DPC Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 1:0 | <p>DPC Trigger Enable - This field enables DPC and controls the conditions that cause DPC to be triggered. Defined encodings are:</p> <ul style="list-style-type: none"> 00b DPC is disabled 01b DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an <u>ERR_FATAL</u> Message 10b DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an <u>ERR_NONFATAL</u> or <u>ERR_FATAL</u> Message 11b Reserved Default value of this field is 00b. | RW |
| 2 | <p>DPC Completion Control - This bit controls the Completion Status for Completions formed during DPC. See § Section 2.9.3 .</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0b Completer Abort (CA) Completion Status 1b Unsupported Request (UR) Completion Status Default value of this bit is 0b. | RW |
| 3 | <p>DPC Interrupt Enable - When Set, this bit enables the generation of an interrupt to indicate that DPC has been triggered. See § Section 6.2.11.1 .</p> <p>Default value of this bit is 0b.</p> | RW |
| 4 | <p>DPC ERR_COR Enable - When Set, this bit enables the sending of an <u>ERR_COR</u> Message to indicate that DPC has been triggered. See § Section 6.2.11.2 .</p> <p>Default value of this bit is 0b.</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 5 | <p>Poisoned TLP Egress Blocking Enable - This bit must be RW if the Poisoned TLP Egress Blocking Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. Software must not Set this bit unless the Poisoned TLP Egress Blocking Supported bit is Set.</p> <p>When Set, this bit enables the associated Egress Port to block the transmission of poisoned TLPs. See § Section 2.7.2.1.</p> <p>Default value of this bit is 0b.</p> | RW / RO |
| 6 | <p>DPC Software Trigger - This bit must be RW if the DPC Software Triggering Supported bit is Set; otherwise, it is permitted to be hardwired to 0b.</p> <p>If DPC is enabled and the DPC Trigger Status bit is Clear, when software writes 1b to this bit, DPC is triggered. Otherwise, software writing a 1b to this bit has no effect.</p> <p>It is permitted to write 1b to this bit while simultaneously writing updated values to other fields in this register, notably the DPC Trigger Enable field. For this case, the DPC Software Trigger semantics are based on the updated value of the DPC Trigger Enable field.</p> <p>This bit always returns 0b when read.</p> | RW / RO |
| 7 | <p>DL_Active ERR_COR Enable - This bit must be RW if the DL_Active ERR_COR Signaling Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. Software must not Set this bit unless the DL_Active ERR_COR Signaling Supported bit is Set.</p> <p>When Set, this bit enables the associated Downstream Port to signal with ERR_COR when the Link transitions to the DL_Active state. See § Section 6.2.11.5.</p> <p>Default value of this bit is 0b.</p> | RW / RO |
| 8 | <p>DPC SIG_SFW Enable - This bit must be implemented if the ERR_COR Subclass Capable bit in the Device Capabilities Register is Set; otherwise, it is permitted to be hardwired to 0b. If the ERR_COR Subclass Capable bit is Clear and software Sets this bit, the behavior is undefined.</p> <p>When Set, this bit enables sending an ERR_COR Message to indicate a DPC event that's been enabled for ERR_COR signaling. See § Section 6.2.11.2 and § Section 6.2.11.5 . This is an additional and alternative way to enable overall DPC ERR_COR signaling beyond the Correctable Error Reporting Enable bit in the Device Control Register . This bit does not affect a Function's ability to send ERR_COR Messages other than the ECS SIG_SFW subclass.</p> <p>Default value of this bit is 0b.</p> | RW / RO |

Base 6.4 vs Base 6.3

7.9.14.4 DPC Status Register (Offset 08h) §

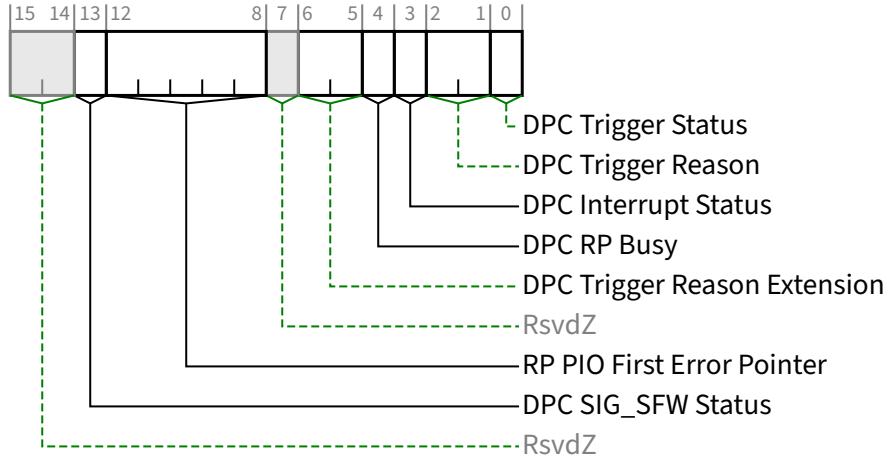


Figure 7-307 DPC Status Register §

Table 7-271 DPC Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>DPC Trigger Status - When Set, this bit indicates that DPC has been triggered, and by definition the Port is “in DPC”. DPC is event triggered.</p> <p>While this bit is Set, hardware must direct the LTSSM to the Disabled State. This bit must be cleared before the LTSSM can be released from the Disabled State, after which the Port is no longer in DPC, and the LTSSM must transition to the Detect State. See § Section 6.2.11 for requirements on how long software must leave the Downstream Port in DPC. Once these requirements are met, software is permitted to clear this bit regardless of the state of other status bits associated with the triggering event.</p> <p>After clearing this bit, software must honor timing requirements defined in § Section 6.6.1 with respect to the first Configuration Read following a Conventional Reset.</p> <p>Default value of this bit is 0b.</p> | RW1CS |
| 2:1 | <p>DPC Trigger Reason - This field indicates why DPC has been triggered. Defined encodings are:</p> <ul style="list-style-type: none"> 00b DPC was triggered due to an unmasked uncorrectable error 01b DPC was triggered due to receiving an ERR_NONFATAL 10b DPC was triggered due to receiving an ERR_FATAL 11b DPC was triggered due to a reason that is indicated by the DPC Trigger Reason Extension field. <p>This field is valid only when the DPC Trigger Status bit is Set; otherwise the value of this field is undefined.</p> | ROS |
| 3 | <p>DPC Interrupt Status - This bit is Set if DPC is triggered while the <u>DPC Interrupt Enable</u> bit is Set. This may cause the generation of an interrupt. See § Section 6.2.11.1 .</p> <p>Default value of this bit is 0b.</p> | RW1CS |
| 4 | <p>DPC RP Busy - When the <u>DPC Trigger Status</u> bit is Set and this bit is Set, the Root Port is busy with internal activity that must complete before software is permitted to Clear the <u>DPC Trigger Status</u> bit. If software Clears the <u>DPC Trigger Status</u> bit while this bit is Set, the behavior is undefined.</p> | RO / RsvdZ |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| | <p>This field is valid only when the <u>DPC Trigger Status</u> bit is Set; otherwise the value of this field is undefined.</p> <p>This bit is applicable only for Root Ports that support <u>RP Extensions for DPC</u>, and is Reserved for Switch Downstream Ports.</p> <p>Default value of this bit is undefined.</p> | |
| 6:5 | <p>DPC Trigger Reason Extension - This field serves as an extension to the DPC Trigger Reason field. When that field is valid and has a value of 11b, this field indicates why DPC has been triggered. Defined encodings are:</p> <ul style="list-style-type: none"> 00b DPC was triggered due to an RP PIO error 01b DPC was triggered due to the DPC Software Trigger bit 10b Reserved 11b Reserved <p>This field is valid only when the <u>DPC Trigger Status</u> bit is Set and the value of the <u>DPC Trigger Reason</u> field is 11b; otherwise the value of this field is undefined.</p> | ROS |
| 12:8 | <p>RP PIO First Error Pointer - The value of this field identifies a bit position in the RP PIO Status Register, and this field is considered valid when that bit is Set. When this field is valid, and software writes a 1b to the indicated RP PIO Status bit (thus clearing it), this field must revert to its default value.</p> <p>This field is applicable only for Root Ports that support <u>RP Extensions for DPC</u>, and otherwise is Reserved. If this field is not Reserved, its default value is 1 1111b, indicating a permanently Reserved RP PIO Status bit, thus guaranteeing that this field is not considered valid.</p> | ROS / RsvdZ |
| 13 | <p>DPC SIG_SFW Status - If the Function supports ERR_COR Subclass capability, this bit must be implemented; otherwise, it must be hardwired to 0b. If implemented, this bit is Set when a SIG_SFW ERR_COR Message is sent to signal a DPC event. See § <u>Section 6.2.11.2</u> and § <u>Section 6.2.11.5</u>.</p> <p>Default value of this bit is 0b</p> | RW1CS / RsvdZ |

7.9.14.5 DPC Error Source ID Register (Offset 0Ah) §

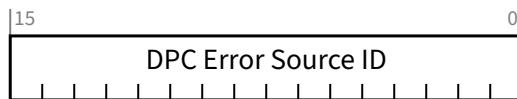


Figure 7-308 DPC Error Source ID Register §

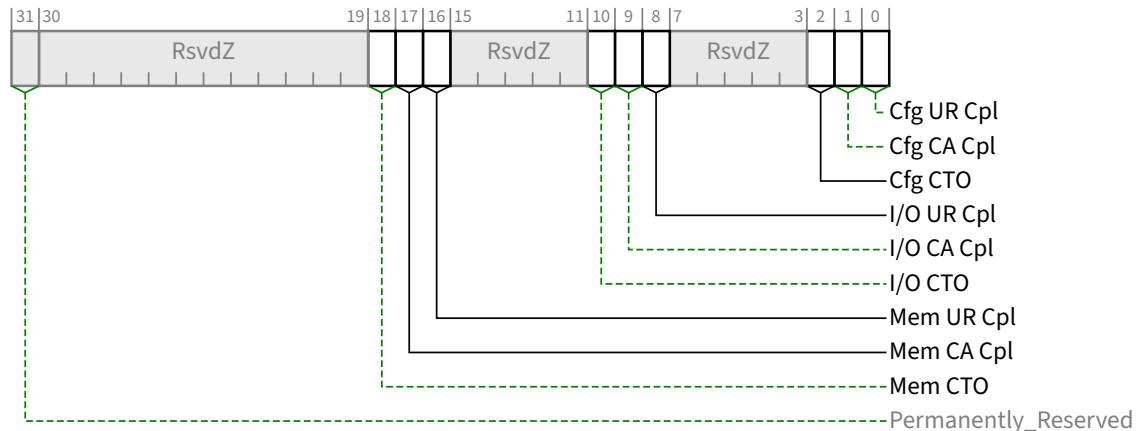
Table 7-272 DPC Error Source ID Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | DPC Error Source ID - When the DPC Trigger Reason field indicates that DPC was triggered due to the reception of an <u>ERR_NONFATAL</u> or <u>ERR_FATAL</u> , this register contains the Requester ID of the received Message. Otherwise, the value of this register is undefined. | ROS |

Base 6.4 vs Base 6.3

7.9.14.6 RP PIO Status Register (Offset 0Ch) §

This register is present only in Root Ports that support RP Extensions for DPC . See § [Section 6.2.11.3](#) .



[Figure 7-309 RP PIO Status Register](#) §

[Table 7-273 RP PIO Status Register](#) §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|------------|---------|
| 0 | Cfg UR Cpl - Configuration Request received UR Completion | RW1CS | 0b |
| 1 | Cfg CA Cpl - Configuration Request received CA Completion | RW1CS | 0b |
| 2 | Cfg CTO - Configuration Request Completion Timeout | RW1CS | 0b |
| 8 | I/O UR Cpl - I/O Request received UR Completion | RW1CS | 0b |
| 9 | I/O CA Cpl - I/O Request received CA Completion | RW1CS | 0b |
| 10 | I/O CTO - I/O Request Completion Timeout | RW1CS | 0b |
| 16 | Mem UR Cpl - Memory Request received UR Completion | RW1CS | 0b |
| 17 | Mem CA Cpl - Memory Request received CA Completion | RW1CS | 0b |
| 18 | Mem CTO - Memory Request Completion Timeout | RW1CS | 0b |
| 31 | Permanently Reserved , since the default RP PIO First Error Pointer field value points to it. | RsvdZ | 0b |

7.9.14.7 RP PIO Mask Register (Offset 10h) §

This register is present only in Root Ports that support RP Extensions for DPC . See § [Section 6.2.11.3](#) .

Base 6.4 vs Base 6.3

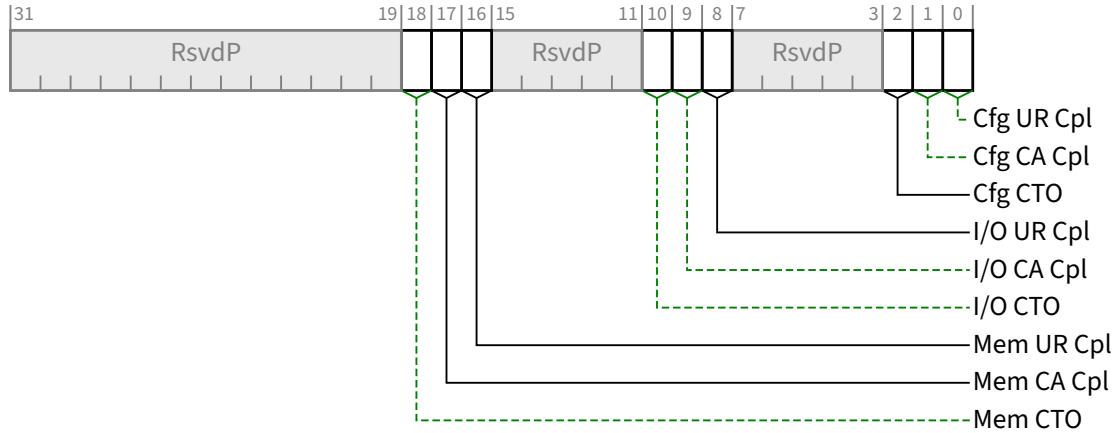


Figure 7-310 RP PIO Mask Register §

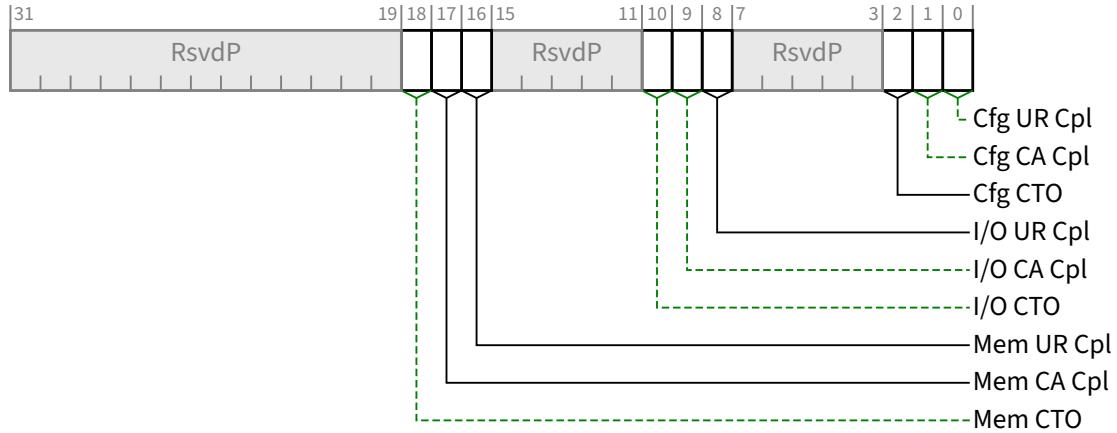
Table 7-274 RP PIO Mask Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|------------|---------|
| 0 | Cfg UR Cpl - Configuration Request received UR Completion | RWS | 1b |
| 1 | Cfg CA Cpl - Configuration Request received CA Completion | RWS | 1b |
| 2 | Cfg CTO - Configuration Request Completion Timeout | RWS | 1b |
| 8 | I/O UR Cpl - I/O Request received UR Completion | RWS | 1b |
| 9 | I/O CA Cpl - I/O Request received CA Completion | RWS | 1b |
| 10 | I/O CTO - I/O Request Completion Timeout | RWS | 1b |
| 16 | Mem UR Cpl - Memory Request received UR Completion | RWS | 1b |
| 17 | Mem CA Cpl - Memory Request received CA Completion | RWS | 1b |
| 18 | Mem CTO - Memory Request Completion Timeout | RWS | 1b |

7.9.14.8 RP PIO Severity Register (Offset 14h) §

This register is present only in Root Ports that support RP Extensions for DPC . See § [Section 6.2.11.3](#) .

Base 6.4 vs Base 6.3

Figure 7-311 RP PIO Severity Register [§](#)Table 7-275 RP PIO Severity Register [§](#)

| Bit Location | Register Description | Attributes | Default |
|--------------|--|------------|---------|
| 0 | Cfg UR Cpl - Configuration Request received UR Completion | RWS | 0b |
| 1 | Cfg CA Cpl - Configuration Request received CA Completion | RWS | 0b |
| 2 | Cfg CTO - Configuration Request Completion Timeout | RWS | 0b |
| 8 | I/O UR Cpl - I/O Request received UR Completion | RWS | 0b |
| 9 | I/O CA Cpl - I/O Request received CA Completion | RWS | 0b |
| 10 | I/O CTO - I/O Request Completion Timeout | RWS | 0b |
| 16 | Mem UR Cpl - Memory Request received UR Completion | RWS | 0b |
| 17 | Mem CA Cpl - Memory Request received CA Completion | RWS | 0b |
| 18 | Mem CTO - Memory Request Completion Timeout | RWS | 0b |

7.9.14.9 RP PIO SysError Register (Offset 18h) [§](#)

This register is present only in Root Ports that support RP Extensions for DPC . See [§](#) [Section 6.2.11.3](#) .

Base 6.4 vs Base 6.3

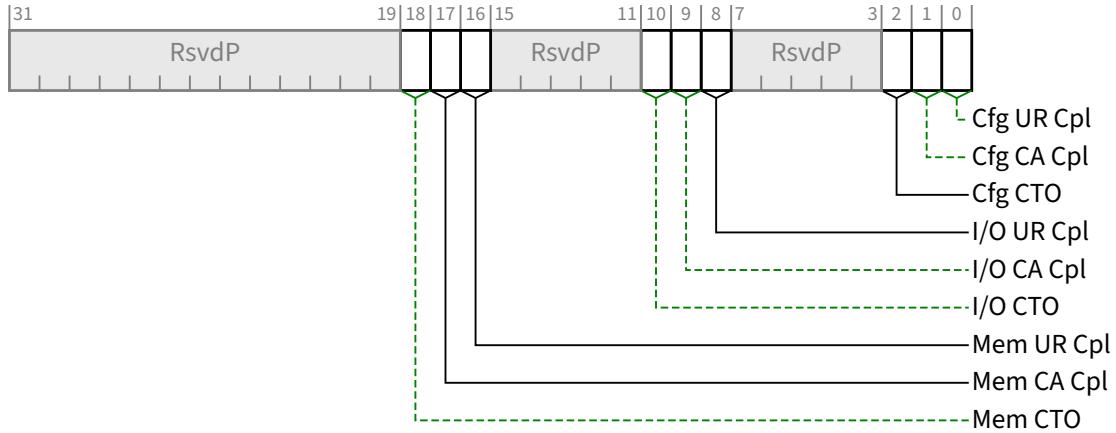


Figure 7-312 RP PIO SysError Register §

Table 7-276 RP PIO SysError Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|------------|---------|
| 0 | Cfg UR Cpl - Configuration Request received UR Completion | RWS | 0b |
| 1 | Cfg CA Cpl - Configuration Request received CA Completion | RWS | 0b |
| 2 | Cfg CTO - Configuration Request Completion Timeout | RWS | 0b |
| 8 | I/O UR Cpl - I/O Request received UR Completion | RWS | 0b |
| 9 | I/O CA Cpl - I/O Request received CA Completion | RWS | 0b |
| 10 | I/O CTO - I/O Request Completion Timeout | RWS | 0b |
| 16 | Mem UR Cpl - Memory Request received UR Completion | RWS | 0b |
| 17 | Mem CA Cpl - Memory Request received CA Completion | RWS | 0b |
| 18 | Mem CTO - Memory Request Completion Timeout | RWS | 0b |

7.9.14.10 RP PIO Exception Register (Offset 1Ch) §

This register is present only in Root Ports that support RP Extensions for DPC . See § [Section 6.2.11.3](#) .

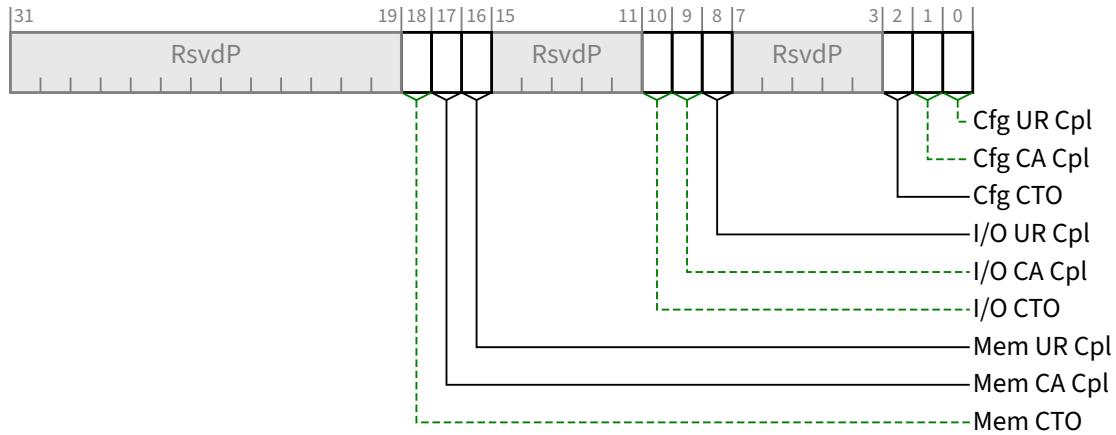


Figure 7-313 RP PIO Exception Register §

Table 7-277 RP PIO Exception Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|------------|---------|
| 0 | Cfg UR Cpl - Configuration Request received UR Completion | RWS | 0b |
| 1 | Cfg CA Cpl - Configuration Request received CA Completion | RWS | 0b |
| 2 | Cfg CTO - Configuration Request Completion Timeout | RWS | 0b |
| 8 | I/O UR Cpl - I/O Request received UR Completion | RWS | 0b |
| 9 | I/O CA Cpl - I/O Request received CA Completion | RWS | 0b |
| 10 | I/O CTO - I/O Request Completion Timeout | RWS | 0b |
| 16 | Mem UR Cpl - Memory Request received UR Completion | RWS | 0b |
| 17 | Mem CA Cpl - Memory Request received CA Completion | RWS | 0b |
| 18 | Mem CTO - Memory Request Completion Timeout | RWS | 0b |

7.9.14.11 RP PIO Header Log Register (Offset 20h) §

This register is implemented only in Root Ports that support RP Extensions for DPC . The RP PIO Header Log Register contains the header from the Request TLP associated with a recorded RP PIO error. Refer to § Section 6.2.11.3 for further details. In Non-Flit Mode, this register is 16 bytes. In Flit Mode, this register is between 52 and 76 bytes and is split into two portions at Offset 20h and Offset 34h. In both Flit Mode and Non-Flit Mode, this register is formatted identically to the Header Log register in AER. See § Section 7.8.4.8 .

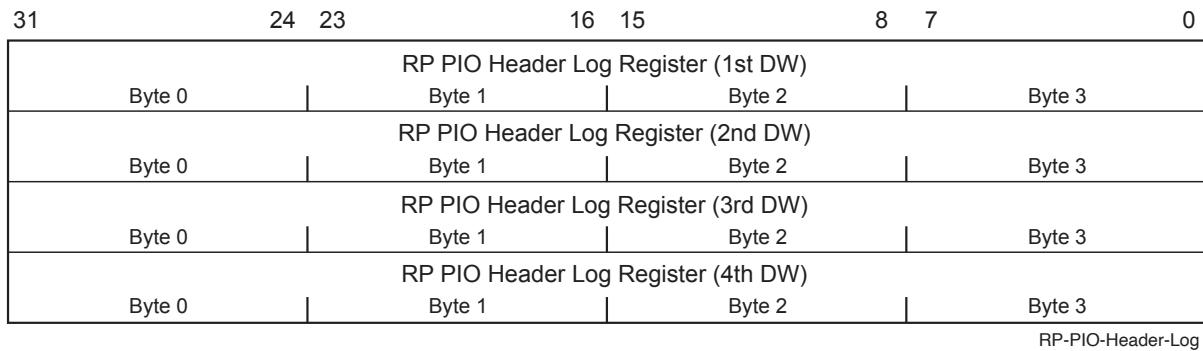


Figure 7-314 RP PIO Header Log Register §

Table 7-278 RP PIO Header Log Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|--|------------|---------|
| 127:0 | TLP Header - of the TLP associated with the error | ROS | 0 |

7.9.14.12 RP PIO ImpSpec Log Register (Offset 30h) §

This register is permitted to be implemented only in Root Ports that support RP Extensions for DPC . The RP PIO ImpSpec Log Register , if implemented, contains implementation specific information associated with the recorded error, e.g., indicating the source of the Request TLP. Space is allocated for this register if the value of the RP PIO Log Size field is 5 or greater. If space is allocated for the register, but the register is not implemented, the bits must be hardwired to 0b.

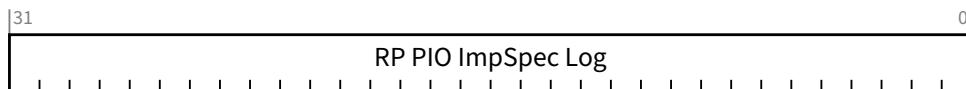


Figure 7-315 RP PIO ImpSpec Log Register §

Table 7-279 RP PIO ImpSpec Log Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|---------------------------|------------|---------|
| 31:0 | RP PIO ImpSpec Log | ROS | 0 |

7.9.14.13 RP PIO TLP Prefix Log Register (Offset 34h) §

This register is permitted to be implemented only in Root Ports that support RP Extensions for DPC .

In Non-Flit Mode, the RP PIO TLP Prefix Log Register contains any End-End TLP Prefixes from the TLP corresponding to a recorded RP PIO error. Refer to § Section 6.2.11.3 for further details.

In Flit Mode, the RP PIO TLP Prefix Log Register does not exist and this configuration space is a continuation of the RP PIO Header Log Register .

If the Root Port supports tracking Non-Posted Requests that contain End-End TLP Prefixes, this register must be implemented, and must be of sufficient size to record the maximum number of End-End TLP Prefixes for any tracked Request. See § [Section 2.9.3](#). The allocated size in DWORDs of the RP PIO TLP Prefix Log Register is the RP PIO Log Size minus 5 if the RP PIO Log Size is 9 or less, or 4 if the RP PIO Log Size is greater than 9. The implemented size of the TLP Prefix Log must be less than or equal to the Root Port's Max End-End TLP Prefixes field value. For the case where the Root Port never transmits Non-Posted Requests containing End-End TLP Prefixes, the allocated and implemented size of the TLP Prefix Log is permitted to be 0. Any DWORDs allocated but not implemented must be hardwired to zero.

This register is formatted identically to the TLP Prefix Log register in AER, although this register's allocated size is variable, whereas the register in AER is always 4 DWORDs. See § [Section 7.8.4.12](#). The First TLP Prefix Log register contains the first End-End TLP Prefix from the TLP, the Second TLP Prefix Log register contains the second End-End TLP Prefix, and so forth. If the TLP contains fewer TLP Prefixes than this register accommodates, any remaining TLP Prefix Log registers must contain zero.

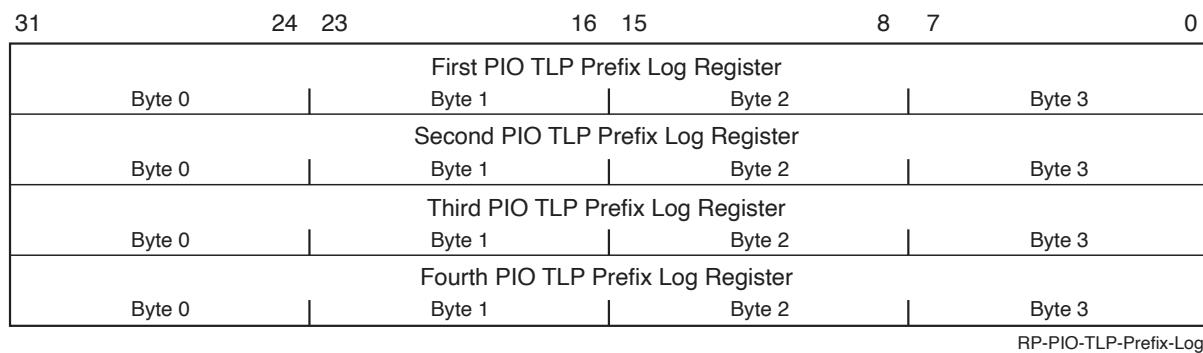


Figure 7-316 RP PIO TLP Prefix Log Register §

Table 7-280 RP PIO TLP Prefix Log Register §

| Bit Location | Register Description | Attributes | Default |
|--------------|------------------------------|------------|---------|
| 127:0 | RP PIO TLP Prefix Log | ROS | 0 |

7.9.15 Precision Time Measurement Extended Capability (PTM Extended Capability) §

The Precision Time Measurement Extended Capability is an optional Extended Capability for discovering and controlling the distribution of a PTM Hierarchy. For Root Complexes, this Capability is required in any Root Port, RCIEP, or RCRB that supports PTM. For Functions associated with an Upstream Port that support PTM, this Capability is required in exactly one Function of that Upstream Port and that Capability controls the PTM behavior of all PTM capable Functions associated with that Upstream Port. For Switch Downstream Ports, PTM behavior is controlled by the same PTM Capability that controls the associated Switch Upstream Port. The PTM Capability is not permitted in Bridges, Switch Downstream Ports, and Root Complex Event Collectors.

For Switches, a single instance of this Capability controls behavior for the entire Switch. If the Upstream Port of the Switch is associated with an MFD, it is not required that the controlling Function be the Function corresponding to the Switch Upstream Port. For a given Switch, if this Capability is present, all Downstream Ports of the Switch must implement the requirements defined in § [Section 6.21.3.2](#).

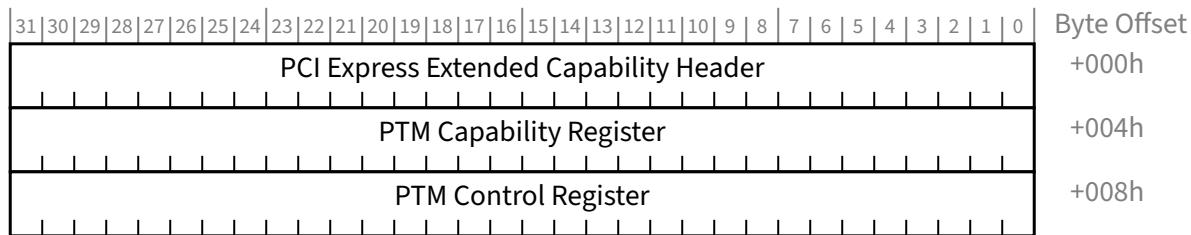


Figure 7-317 PTM Extended Capability Structure §

7.9.15.1 PTM Extended Capability Header (Offset 00h) §

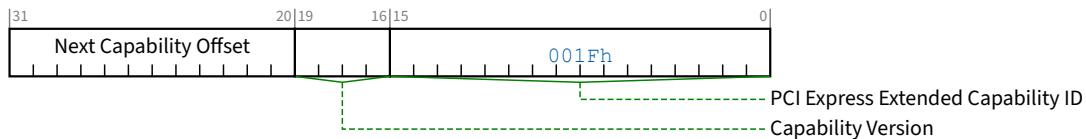


Figure 7-318 PTM Extended Capability Header §

Table 7-281 PTM Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Precision Time Measurement Capability is 001Fh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | RO |

7.9.15.2 PTM Capability Register (Offset 04h) §

This register describes a Function's support for Precision Time Measurement. Not all fields within this register apply to all Functions capable of implementing PTM.

Base 6.4 vs Base 6.3

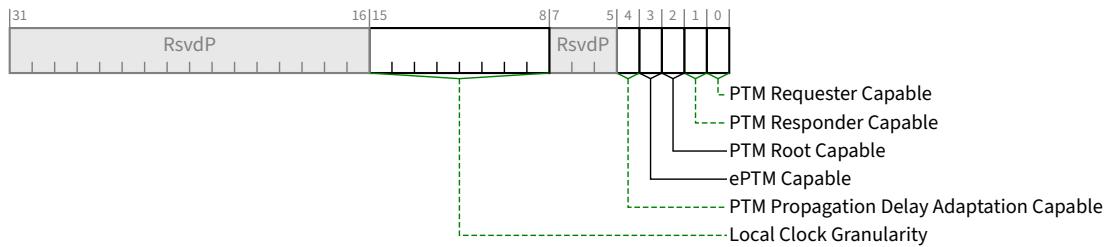


Figure 7-319 PTM Capability Register

Table 7-282 PTM Capability Register

| Bit Location | Register Description | Attributes | | | | | | |
|---------------------------------|---|-------------------|--|---------------------------------|---|-------------------|--|----------------|
| 0 | <p>PTM Requester Capable - Indicates the Function implements the PTM Requester role (see § Section 6.21.3.1).</p> <p>Endpoints and RCiEPs are permitted to Set this bit to indicate that they implement the PTM Requester role.</p> <p>Switch Upstream Ports must Set this bit if the Switch contains one or more of the following:</p> <ul style="list-style-type: none"> • A Downstream Port that implements the PTM Responder role. • An additional Function that implements the PTM Requester role. | HwInit | | | | | | |
| 1 | <p>PTM Responder Capable - Root Ports and RCRBs are permitted to, and Switches supporting PTM must, Set this bit to indicate they implement the PTM Responder role (see § Section 6.21.3.2).</p> <p>If PTM Root Capable is Set, then this bit must be Set.</p> | HwInit | | | | | | |
| 2 | <p>PTM Root Capable - Root Ports, RCRBs , and Switches are permitted to Set this bit if they are capable of being a source of PTM Master Time (see § Section 6.21.1).</p> <p>All other Functions must hardwire this bit to 0b.</p> | HwInit | | | | | | |
| 3 | <p>ePTM Capable - If Set, indicates that this device supports Enhanced Precision Time Measurement (ePTM). This bit MUST be Set in all PTM Devices.</p> | HwInit | | | | | | |
| 4 | <p>PTM Propagation Delay Adaptation Capable – When Set, this field indicates the Port supports the PTM Propagation Delay Adaptation Capability , controlled via the PTM Propagation Delay Adaptation Interpretation B bit in the Link Control Register . For a Switch, when Set in the Upstream Port of the Switch, indicates that the Upstream Port and all Downstream Ports of the Switch support the PTM Propagation Delay Adaptation Capability , controlled per Port via the PTM Propagation Delay Adaptation Interpretation B bit in the Link Control Register of each Port.</p> | HwInit | | | | | | |
| 15:8 | <p>Local Clock Granularity - Encodings are:</p> <table border="0"> <tr> <td>0000 0000b</td><td>Time Source does not implement a local clock. It simply propagates timing information obtained from further Upstream in the PTM Hierarchy when responding to PTM Request messages.</td></tr> <tr> <td>0000 0001b to 1111 1110b</td><td>Indicates the period of this Time Source's local clock in ns.</td></tr> <tr> <td>1111 1111b</td><td>Indicates the period of this Time Source's local clock is greater than 254 ns.</td></tr> </table> | 0000 0000b | Time Source does not implement a local clock. It simply propagates timing information obtained from further Upstream in the PTM Hierarchy when responding to PTM Request messages. | 0000 0001b to 1111 1110b | Indicates the period of this Time Source's local clock in ns. | 1111 1111b | Indicates the period of this Time Source's local clock is greater than 254 ns. | HwInit / RsvdP |
| 0000 0000b | Time Source does not implement a local clock. It simply propagates timing information obtained from further Upstream in the PTM Hierarchy when responding to PTM Request messages. | | | | | | | |
| 0000 0001b to 1111 1110b | Indicates the period of this Time Source's local clock in ns. | | | | | | | |
| 1111 1111b | Indicates the period of this Time Source's local clock is greater than 254 ns. | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | <p>If the PTM Root Select bit is Set, this local clock is used to provide PTM Master Time. Otherwise, the Time Source uses this local clock to locally track PTM Master Time received from further Upstream within a PTM Hierarchy.</p> <p>This field is RsvdP if the PTM Root Capable bit is 0b.</p> | |

7.9.15.3 PTM Control Register (Offset 08h) §

This register controls a Function's participation in the Precision Time Measurement mechanism. Not all fields within this register apply to all Functions capable of implementing PTM.

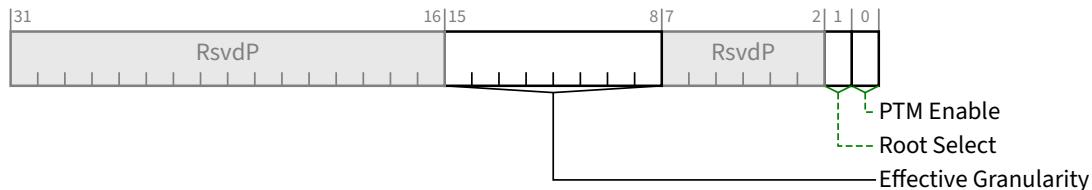


Figure 7-320 PTM Control Register §

Table 7-283 PTM Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>PTM Enable - When Set, this Function is permitted to participate in the PTM mechanism according to its selected role(s) (see § Section 6.21.2).</p> <p>Default value is 0b.</p> | RW |
| 1 | <p>Root Select - When Set, if the PTM Enable bit is also Set, this Time Source is the PTM Root.</p> <p>Within each PTM Hierarchy, it is recommended that system software select only the furthest Upstream Time Source to be the PTM Root.</p> <p>Default value is 0b. If the value of the PTM Root Capable bit is 0b, this bit is permitted to be hardwired to 0b.</p> | RW / RO |
| 15:8 | <p>Effective Granularity - For Functions implementing the PTM Requester Role, this field provides information relating to the expected accuracy of the PTM clock, but does not otherwise affect the PTM mechanism.</p> <p>For Endpoints, system software must program this field to the value representing the maximum Local Clock Granularity reported by the PTM Root and all intervening PTM Time Sources.</p> <p>For RCiEPs, system software must set this field to the value reported in the Local Clock Granularity field by the associated PTM Time Source.</p> <p>Permitted values:</p> <ul style="list-style-type: none"> 0000 0000b Unknown PTM granularity - one or more Switches between this Function and the PTM Root reported a Local Clock Granularity value of 0000 0000b. 0000 0001b to 1111 1110b Indicates the effective PTM granularity in ns. | RW / RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | 1111 1111b Indicates the effective PTM granularity is greater than 254 ns. Default value is 0000 0000b. If <u>PTM Requester Capable</u> is Clear, this field is permitted to be hardwired to 0000 0000b. | |

7.9.16 Readiness Time Reporting Extended Capability §

The Readiness Time Reporting Extended Capability provides an optional mechanism for describing the time required for a Device or Function to become Configuration-Ready. In the indicated situations, software is permitted to issue Requests to the Device or Function after waiting for the time advertised in this capability and need not wait for the (longer) times required elsewhere.

Software is permitted to issue requests upon the earliest of:

- Receiving a Readiness Notifications message (see § Section 6.22).
- Waiting the appropriate time as specified in this document or in applicable specifications including the [PCI] and the [PCI-PM].
- Waiting the time indicated in the associated field of this capability.
- Waiting the time defined by system software or firmware²⁰⁴ .

Software is permitted to cache values from this capability and to use those cached values as long as the same device operating in the same manner has not changed.

This capability is permitted to be implemented in all Functions.

The capability is optional for PFs and VFs. However, if a VF associated with a given PF contains the capability, all VFs associated with that PF must contain the capability and report the same time values.

For VFs, see § Section 5.10.1). Other Functions must be Configuration-Ready if:

- The Immediate Readiness bit is Clear and at least Reset Time has elapsed after the completion of Conventional Reset
 - If the Immediate Readiness bit is Set, Reset Time does not apply, and is Reserved
- The Function is associated with an Upstream Port and at least DL_Up Time has elapsed after the Downstream Port above that Function reported Data Link Layer Link Active (see § Section 7.5.3.8).
- The Function supports Function Level Reset and at least FLR Time has elapsed after that Function was issued a Function Level Reset.
- Immediate_Readiness_on_Return_to_D0 is Clear and at least D3_Hot to D0 Time has elapsed after that Function was directed to the D0 state from D3_Hot.
 - If the Immediate_Readiness_on_Return_to_D0 bit is Set, D3_Hot to D0 Time does not apply, and is Reserved

When Immediate_Readiness_on_Return_to_D0 is Clear, a Function must be Configuration-Ready when at least D3_Hot to D0 Time has elapsed after the Function was directed to the D0 state from D3_Hot. In addition, the Function must be in either the D0_uninitialized or D0_active state, depending on the value of the No_Soft_Reset bit.

If the above conditions do not apply, Function behavior is not determined by the Readiness Time Reporting Extended Capability, and the Function must respond as defined elsewhere (including, for example, no response or a response with Configuration Retry Status).

204. For example, using ACPI tables to provide the equivalent of this capability.

The time values reported are determined by implementation specific mechanisms. A Valid bit is defined in this capability to permit a device to defer reporting time values, for example to allow hardware initialization through driver-based mechanisms. If the Valid bit remains Clear and 1 minute has elapsed after device driver(s) have started, software is permitted to assume that no values will be reported.

Registers and fields in the Readiness Time Reporting Extended Capability are shown in § Figure 7-321 . Time values are encoded in floating point as shown in § Figure 7-322 . The actual time value is $\text{Value} \times \text{Multiplier}[\text{Scale}]$. For example, the value A1Eh represents about 1 second (actually 1.006 sec) and the value 80Ah represents about 10 ms (actually 10.240 ms).

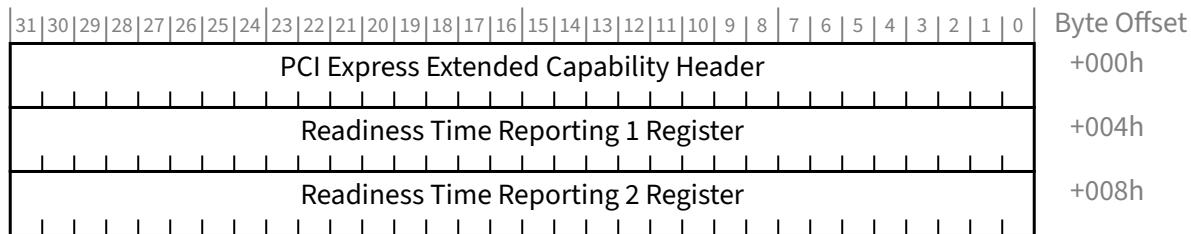


Figure 7-321 Readiness Time Reporting Extended Capability §

| Scale | Multiplier |
|-------|---------------|
| 0 | 1 ns |
| 1 | 32 ns |
| 2 | 1,024 ns |
| 3 | 32,768 ns |
| 4 | 1,048,576 ns |
| 5 | 33,554,432 ns |
| 6 | Reserved |
| 7 | Reserved |

Multiplier = 32^{Scale}

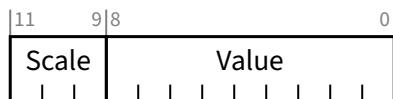


Figure 7-322 Readiness Time Encoding §

7.9.16.1 Readiness Time Reporting Extended Capability Header (Offset 00h) §

§ Figure 7-323 and § Table 7-285 detail allocation of fields in the Extended Capability header.

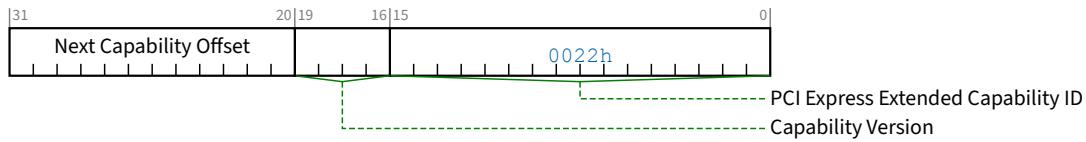


Table 7-285 Readiness Time Reporting Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Readiness Time Reporting Extended Capability is 0022h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.9.16.2 Readiness Time Reporting 1 Register (Offset 04h) §

§ Figure 7-324 and § Table 7-286 detail allocation of fields in the Readiness Time Reporting 1 Register.

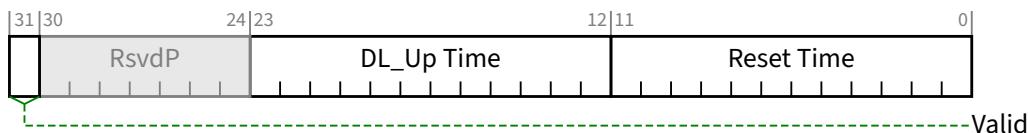


Table 7-286 Readiness Time Reporting 1 Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------|
| 11:0 | Reset Time - is the time a non-VF Function requires to become Configuration-Ready after the completion of Conventional Reset. For VF semantics, see § Section 9.4.3.3.1 . This field is RsvdP if the Immediate Readiness bit is Set. This field is undefined when the Valid bit is Clear. This field must be less than or equal to the encoded value A1Eh. | HwInit / RsvdP |

| Bit Location | Register Description | Attributes |
|--------------|--|-----------------------------------|
| 23:12 | <p>DL Up Time - is the time the Function requires to become Configuration-Ready after the Downstream Port above the Function reports Data Link Layer Link Active .</p> <p>This field is RsvdP in Functions that are not associated with an Upstream Port.</p> <p>For VFs, this field is not applicable and is RsvdP .</p> <p>This field is undefined when the Valid bit is Clear.</p> <p>This field must be less than or equal to the encoded value A1Eh.</p> | HwInit / RsvdP VF RsvdP |
| 31 | <p>Valid - If Set, indicates that all time values in this capability are valid. If Clear, indicates that the time values in this capability are not yet available.</p> <p>Time values may depend on device configuration. Device specific mechanisms, possibly involving the device driver(s), could be involved in determining time values.</p> <p>If this bit remains Clear and 1 minute has elapsed after all associated device driver(s) have started, software is permitted to assume that this bit will never be set.</p> | HwInit |

7.9.16.3 Readiness Time Reporting 2 Register (Offset 08h) §

§ Figure 7-325 and § Table 7-287 detail allocation of fields in the Readiness Time Reporting 2 Register.

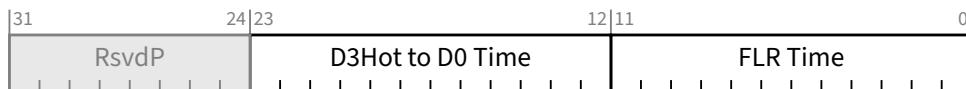


Figure 7-325 Readiness Time Reporting 2 Register §

Table 7-287 Readiness Time Reporting 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------|
| 11:0 | <p>FLR Time - is the time that the Function requires to become Configuration-Ready after it was issued an FLR .</p> <p>This field is RsvdP when the Function Level Reset Capability bit is Clear (see § Section 7.5.3.3).</p> <p>This field is undefined when the Valid bit is Clear.</p> <p>This field must be less than or equal to the encoded value A1Eh.</p> | HwInit / RsvdP |
| 23:12 | <p>D3Hot to D0 Time - If Immediate_Readiness_on_Return_to_D0 is Clear, D3Hot to D0 Time is the time that a non-VF Function requires after it is directed from D3Hot to D0 before it is Configuration-Ready and has returned to either D0uninitialized or D0active state. For VF semantics, see § Section 5.10.1 .</p> <p>This field is RsvdP if the Immediate_Readiness_on_Return_to_D0 bit is Set.</p> <p>For a VF that does not implement the PCI Power Management Capability , this field is undefined.</p> <p>This field is undefined when the Valid bit is Clear.</p> <p>This field must be less than or equal to the encoded value 80Ah.</p> | HwInit / RsvdP |

7.9.17 Hierarchy ID Extended Capability §

The Hierarchy ID Extended Capability provides an optional mechanism for passing a unique identifier to Functions within a Hierarchy. At most one instance of this capability is permitted in a Function. This capability is not applicable to Bridges, Root Complex Event Collectors, and RCRBs.

This capability takes three forms:

In Upstream Ports:

- This capability is permitted any Function associated with an Upstream Port.
- This capability is optional in Switch Upstream Ports. Support in Switch Upstream and Downstream Ports is independently optional.
- This capability is mandatory in Functions that use the Hierarchy ID Message. This includes use by the Function's driver.
- Functions, other than VFs, that have Hierarchy ID Writeable Clear, must report the Message Requester ID, Hierarchy ID, System GUID Authority ID, and System GUID fields from the most recently received Hierarchy ID Message.
- All VFs that have Hierarchy ID Writeable Clear, must report the same Hierarchy ID Valid, Message Requester ID, Hierarchy ID, System GUID Authority ID, and System GUID values as their associated PF.
- PFs must implement this capability if any of their VFs implement this capability.
- Functions that have Hierarchy ID Writeable Set must report the Hierarchy ID Valid, Message Requester ID, Hierarchy ID, System GUID Authority ID, and System GUID values programmed by software.

In Downstream Ports:

- This capability is permitted in any Downstream Port. It is recommended that it be implemented in Root Ports.
- When present in a Switch Downstream Port, this capability must be implemented in all Downstream Ports of the Switch. Support in Switch Upstream and Downstream Ports is independently optional.
- In Downstream Ports, the Hierarchy ID, System GUID Authority ID, and System GUID fields are Read / Write and contain the values to send in the Hierarchy ID Message.
- A Hierarchy ID capability is not affected by Hierarchy ID Messages forwarded through the associated Downstream Port.

In RCiEPs:

- VFs that have Hierarchy ID Writeable Clear must report the same Message Requester ID, Hierarchy ID, System GUID Authority ID, and System GUID values as their associated PF.
- PFs must implement this capability if any of their VFs implement this capability.
- Functions, other than VFs, that have Hierarchy ID Writeable Clear, must report the same Hierarchy ID Valid, Message Requester ID, Hierarchy ID, System GUID Authority ID, and System GUID values. The source of this information is outside the scope of this specification.
- Functions that have Hierarchy ID Writeable Set must report the Hierarchy ID Valid, Message Requester ID, Hierarchy ID, System GUID Authority ID, and System GUID values programmed by software.

§ Figure 7-326 details the layout of the Hierarchy ID Extended Capability.

Base 6.4 vs Base 6.3

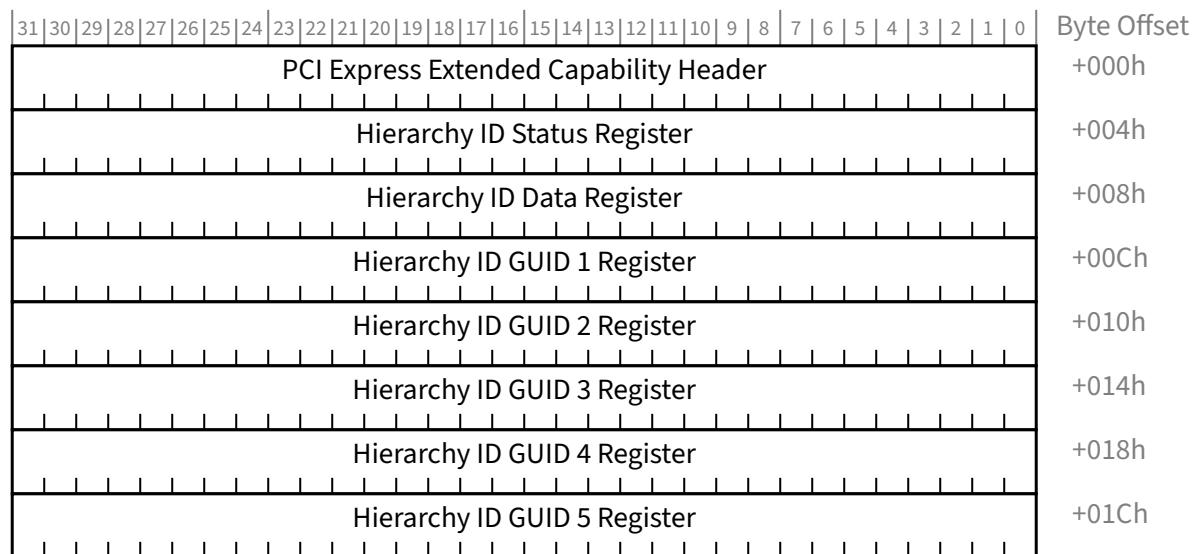


Figure 7-326 Hierarchy ID Extended Capability [§](#)

7.9.17.1 Hierarchy ID Extended Capability Header (Offset 00h) [§](#)

[§ Figure 7-327](#) and [§ Table 7-288](#) detail allocation of fields in the Hierarchy ID Extended Capability Header.



Figure 7-327 Hierarchy ID Extended Capability Header [§](#)

Table 7-288 Hierarchy ID Extended Capability Header [§](#)

| Bit Location | Description | Attributes |
|--------------|--|------------|
| 15:0 | Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Hierarchy ID Extended Capability is 0028h. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities in configuration space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating the list of Capabilities) or greater than OFFh. | RO |

7.9.17.2 Hierarchy ID Status Register (Offset 04h) §

§ Figure 7-328 and § Table 7-289 detail allocation of fields in the Hierarchy ID Status Register .

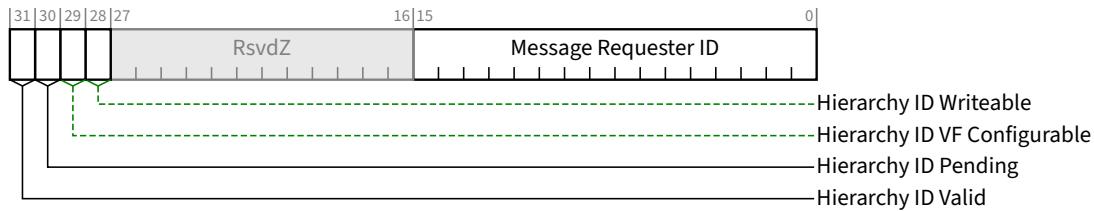


Figure 7-328 Hierarchy ID Status Register §

Table 7-289 Hierarchy ID Status Register §

| Bit Location | Description | Attributes |
|--------------|--|---------------|
| 15:0 | <p>Message Requester ID - In an Upstream Port, this field contains the Requester ID from the most recently received Hierarchy ID Message . This field is meaningful only if Hierarchy ID Valid is 1b. This value identifies the Downstream Port (within this Hierarchy) that sent the Hierarchy ID Message. This information is not considered part of the Hierarchy ID as it can vary within the Hierarchy (e.g., different Root Ports of one Root Complex), but helps in debug situations to identify the provenance of the Hierarchy ID information.</p> <p>In a Downstream Port, this field is RsvdZ .</p> <p>For RCiEPs , this field is RsvdZ .</p> <p>This field defaults to 0000h.</p> | RO / RsvdZ |
| 28 | <p>Hierarchy ID Writeable - This bit is Set to indicate that the Hierarchy ID Data and GUID registers are read/write. This bit is Clear to indicate that the Hierarchy ID and GUID registers are read only.</p> <p>In Downstream Ports this bit is hardwired to 1b.</p> <p>In Upstream Ports, Functions that are not VFs must hardwire this bit to 0b.</p> <p>RCiEPs that are not VFs, must hardwire this bit to either 0b or 1b.</p> <p>VFs in an Upstream Port and Root Complex Integrated VFs are permitted to either:</p> <ul style="list-style-type: none"> • hardwire this bit to 0b or • implement this bit as read / write with a default value of 0b. | RW / RO |
| 29 | <p>Hierarchy ID VF Configurable - This bit indicates that Hierarchy ID Writeable can be configured.</p> <p>If Hierarchy ID Writeable is implemented as read / write, this bit is 1b. Otherwise this bit is 0b.</p> | RO |
| 30 | <p>Hierarchy ID Pending - In Downstream Ports this requests the ↓↑transmittion↓ ↑↑transmission↑ of a Hierarchy ID Message . Setting it requests transmission of a message based on the Hierarchy Data and GUID registers in this capability. This bit is cleared when either the transmit request is satisfied or the Link enters DL_Down . Behavior is undefined if the Hierarchy Data or GUID registers in this capability are written while this bit is Set.</p> <p>In Downstream Ports, this bit is Read / Write defaulting to 0b.</p> <p>In all other Functions, this bit is RsvdZ .</p> | RW / RsvdZ |

| Bit Location | Description | Attributes |
|--------------|---|----------------|
| 31 | <p>Hierarchy ID Valid - This bit indicates that the remaining fields in this capability are meaningful.</p> <p>In Downstream Ports, this bit is hardwired to 1b.</p> <p>In all other Functions, the following rules apply:</p> <ul style="list-style-type: none"> If <u>Hierarchy ID Writeable</u> is Set, this bit is read/write, default 0b. If <u>Hierarchy ID Writeable</u> is Clear, this bit is read only, default 0b. <ul style="list-style-type: none"> In VFs, this bit contains the same value as the associated PF. In Functions other than VFs that are associated with an Upstream Port, this bit is Set when a <u>Hierarchy ID Message</u> is received, and Cleared when the Link is <u>DL_Down</u>. In RCiEPs other than VFs, this bit contains a system provided value. The mechanism for determining this value is outside the scope of this specification. | <u>RW / RO</u> |

7.9.17.3 Hierarchy ID Data Register (Offset 08h) §

§ Figure 7-329 and § Table 7-290 detail allocation of fields in the Hierarchy ID Data Register .

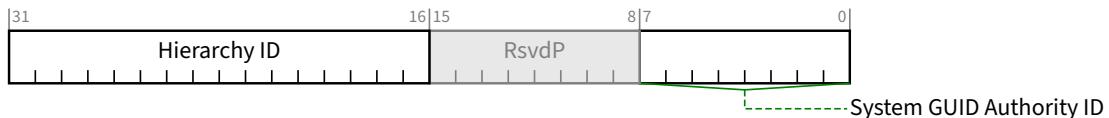


Figure 7-329 Hierarchy ID Data Register §

Table 7-290 Hierarchy ID Data Register §

| Bit Location | Description | Attributes |
|--------------|---|----------------|
| 7:0 | <p>System GUID Authority ID - This field corresponds to the <u>System GUID Authority ID</u> field in the <u>Hierarchy ID Message</u>. See § Section 6.25 for details.</p> <p>This field is meaningful only if <u>Hierarchy ID Valid</u> is 1b.</p> <p>If <u>Hierarchy ID Writeable</u> is Set, this field is read-write and contains the value programmed by software.</p> <p>If <u>Hierarchy ID Writeable</u> is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 00h.</p> | <u>RO / RW</u> |
| 31:16 | <p>Hierarchy ID - This field corresponds to the <u>Hierarchy ID</u> field in the <u>Hierarchy ID Message</u>. See § Section 6.25 for details.</p> <p>This field is meaningful only if <u>Hierarchy ID Valid</u> is 1b.</p> <p>If <u>Hierarchy ID Writeable</u> is Set, this field is read-write and contains the value programmed by software.</p> <p>If <u>Hierarchy ID Writeable</u> is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 0000h.</p> | <u>RO / RW</u> |

7.9.17.4 Hierarchy ID GUID 1 Register (Offset 0Ch) §

§ Figure 7-330 and § Table 7-291 detail allocation of fields in the Hierarchy ID GUID 1 Register .

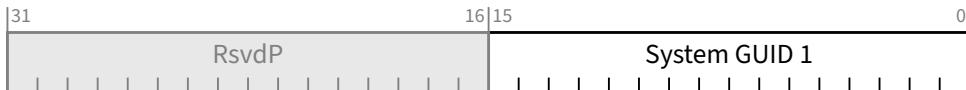


Figure 7-330 Hierarchy ID GUID 1 Register §

Table 7-291 Hierarchy ID GUID 1 Register §

| Bit Location | Description | Attributes |
|--------------|---|------------|
| 15:0 | <p>System GUID 1 - This field corresponds to bits [143:128] of the System GUID in the Hierarchy ID Message . See § Section 6.25 for details.</p> <p>This field is meaningful only if Hierarchy ID Valid is 1b.</p> <p>If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software.</p> <p>If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 0000h.</p> | RO / RW |

7.9.17.5 Hierarchy ID GUID 2 Register (Offset 10h) §

§ Figure 7-331 and § Table 7-292 detail allocation of fields in the Hierarchy ID GUID 2 Register .

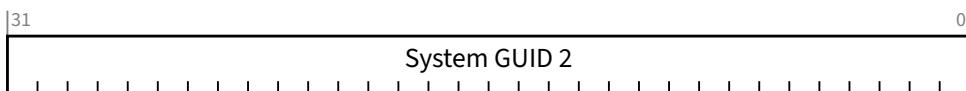


Figure 7-331 Hierarchy ID GUID 2 Register §

Table 7-292 Hierarchy ID GUID 2 Register §

| Bit Location | Description | Attributes |
|--------------|---|------------|
| 31:0 | <p>System GUID 2 - This field corresponds to bits [127:96] of the System GUID field in the Hierarchy ID Message . See § Section 6.25 for details.</p> <p>This field is meaningful only if Hierarchy ID Valid is 1b.</p> <p>If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software.</p> <p>If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 0000 0000h.</p> | RO / RW |

7.9.17.6 Hierarchy ID GUID 3 Register (Offset 14h) §

§ Figure 7-332 and § Table 7-293 detail allocation of fields in the Hierarchy ID GUID 3 Register .

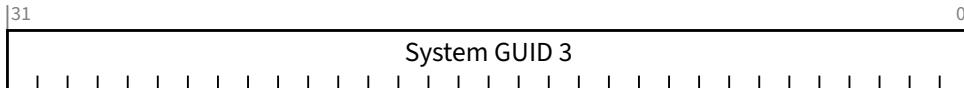


Figure 7-332 Hierarchy ID GUID 3 Register §

Table 7-293 Hierarchy ID GUID 3 Register §

| Bit Location | Description | Attributes |
|--------------|--|------------|
| 31:0 | <p>System GUID 3 - This field corresponds to bits [95:64] of the System GUID field in the Hierarchy ID Message . See § Section 6.25 for details.</p> <p>This field is meaningful only if Hierarchy ID Valid is 1b.</p> <p>If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software.</p> <p>If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 0000 0000h.</p> | RO / RW |

7.9.17.7 Hierarchy ID GUID 4 Register (Offset 18h) §

§ Figure 7-333 and § Table 7-294 detail allocation of fields in the Hierarchy ID GUID 4 Register .

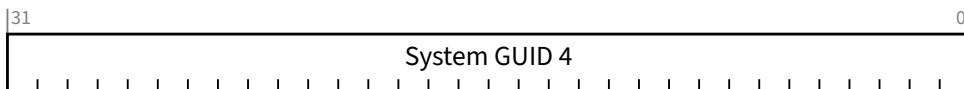


Figure 7-333 Hierarchy ID GUID 4 Register §

Table 7-294 Hierarchy ID GUID 4 Register §

| Bit Location | Description | Attributes |
|--------------|--|------------|
| 31:0 | <p>System GUID 4 - This field corresponds to bits [63:32] of the System GUID field in the Hierarchy ID Message . See § Section 6.25 for details.</p> <p>This field is meaningful only if Hierarchy ID Valid is 1b.</p> <p>If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software.</p> <p>If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 0000 0000h.</p> | RO / RW |

7.9.17.8 Hierarchy ID GUID 5 Register (Offset 1Ch) §

§ Figure 7-334 and § Table 7-295 detail allocation of fields in the Hierarchy ID GUID 5 Register .

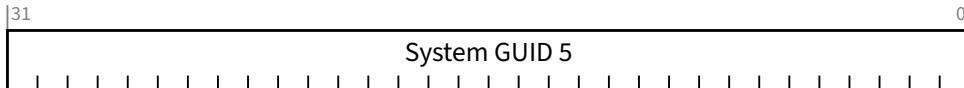


Figure 7-334 Hierarchy ID GUID 5 Register §

Table 7-295 Hierarchy ID GUID 5 Register §

| Bit Location | Description | Attributes |
|--------------|---|------------|
| 31:0 | <p>System GUID 5 - This field corresponds to bits [31:0] of the System GUID field in the Hierarchy ID Message . See § Section 6.25 for details.</p> <p>This field is meaningful only if Hierarchy ID Valid is 1b.</p> <p>If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software.</p> <p>If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in § Section 7.9.17 .</p> <p>This field defaults to 0000 0000h.</p> | RO / RW |

7.9.18 Vital Product Data Capability (VPD Capability) §

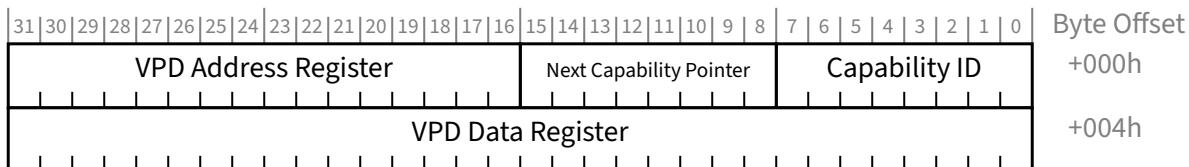
Support of VPD is optional. All Functions are permitted to contain the capability. This includes all Functions of a Multi-Function Device associated with an Upstream Port as well as RCIEPs . This also includes PFs and VFs.

Vital Product Data (VPD) is information that uniquely identifies hardware and, potentially, software elements of a system. The VPD can provide the system with information on various Field Replaceable Units such as part number, serial number, and other detailed information. The objective from a system point of view is to make this information available to the system owner and service personnel. VPD typically resides in a storage device (for example, a serial EEPROM) associated with the Function.

VFs and PFs that implement the VPD Capability must ensure that there can be no “data leakage” between VFs and/or PFs via the VPD Capability.

Details of the VPD Data is defined in § Section 6.27 .

Access to the VPD is provided using the Capabilities List in Configuration Space. The VPD Capability structure is shown in § Figure 7-335 .

*Figure 7-335 VPD Capability Structure* §

The following protocols are used transfer data between the VPD Data field and the VPD storage component.

- To read VPD information:
 1. Issue single write to the VPD Address Register writing the flag bit (F) to 0b and VPD Address with the address to read.
 2. The hardware device will set F to 1b when 4 bytes of data from the storage component have been transferred to VPD Data.
 3. Software can monitor F and, after it becomes 1b, read the VPD information from VPD Data.

Behavior is undefined if either the VPD Address or VPD Data is written, prior to the flag bit becoming 1b.

- To write VPD information to the read/write portion of the VPD space:
 1. Write the data to VPD Data
 2. Then issue a single write to the VPD Address Register with F set to 1b and VPD Address set to the address where the VPD Data is to be stored.
 3. The software then monitors F and when it is set to 0b (by device hardware), the VPD Data (all 4 bytes) has been transferred from VPD Data to the storage component.

If either the VPD Address or VPD Data is written, prior to F being becoming 0b, the results of the write operation to the storage component are unpredictable.

Behavior is undefined if a read or write of the storage component is requested and VPD Address is outside the range of the storage component.

The VPD (both the read only items and the read/write fields) is stored information and will have no direct control of any device operations.

7.9.18.1 VPD Address Register §

The VPD Address Register is used to request a read or write of the VPD storage component.

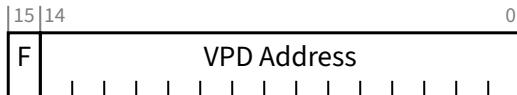


Figure 7-336 VPD Address Register §

Table 7-296 VPD Address Register §

| Bit Location | Description | Attributes |
|--------------|--|--|
| 14:0 | VPD Address - DWORD-aligned byte address of the VPD to be accessed. Behavior is undefined if the lowest 2 bits of this field are non-zero. The lowest two bits of the field must be either <u>RW</u> , or <u>RO</u> with a value of 00b. The remaining bits of the field must be <u>RW</u> . Default is implementation specific. | <u>RW</u> / <u>RO</u> (see description) |
| 15 | F - The F bit is always written along with VPD Address . The value of F indicates the direction of transfer being requested (0b = read, 1b = write). When the transfer is complete, the F bit value changes to indicate completion (1b = read complete, 0b = write complete). Default is implementation specific. | <u>RW</u> |

7.9.18.2 VPD Data Register §

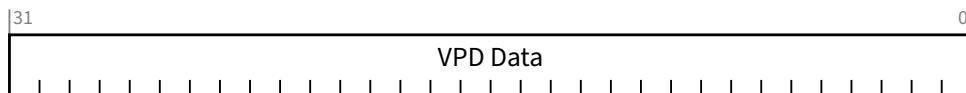


Figure 7-337 VPD Data Register §

Table 7-297 VPD Data Register §

| Bit Location | Description | Attributes |
|--------------|--|------------|
| 31:0 | VPD Data - VPD Data can be read through this register. The least significant byte of this register (at offset 04h in this capability structure) corresponds to the byte of VPD at the address specified by <u>VPD Address</u> . Behavior is undefined for any read or write of this register with Byte Enables other than 1111b. Default is implementation specific. | <u>RW</u> |

7.9.19 Native PCIe Enclosure Management Extended Capability (NPEM Extended Capability) §

The Native PCIe Enclosure Management Extended (NPEM) Capability is an optional extended capability that is permitted to be implemented by Root Ports, Switch Downstream Ports, and Endpoints.

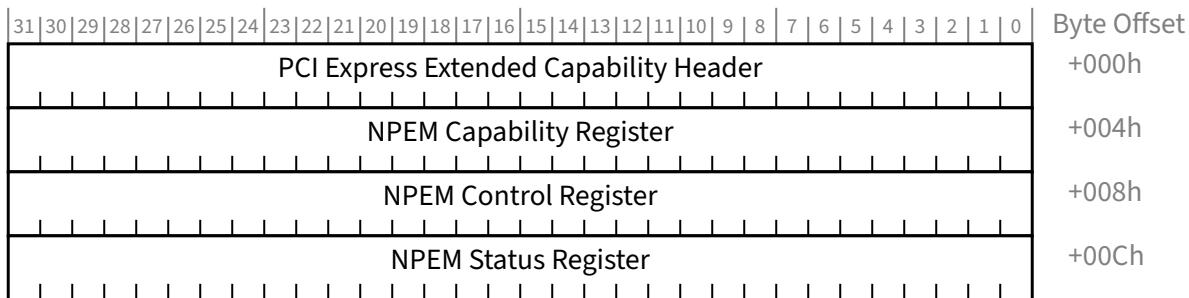


Figure 7-338 NPEM Extended Capability §

7.9.19.1 NPEM Extended Capability Header (Offset 00h) §

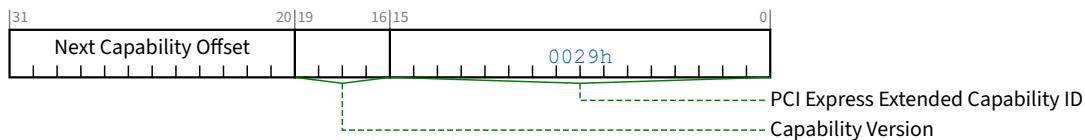


Figure 7-339 NPEM Extended Capability Header §

Table 7-298 NPEM Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the NPEM Extended Capability is 0029h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

7.9.19.2 NPEM Capability Register (Offset 04h) §

The NPEM Capability Register contains an overall NPEM Capable bit and a bit map of states supported in the implementation. Implementations are required to support OK, Locate, Fail, and Rebuild states if NPEM Capable bit is Set. All other states are optional.

Base 6.4 vs Base 6.3

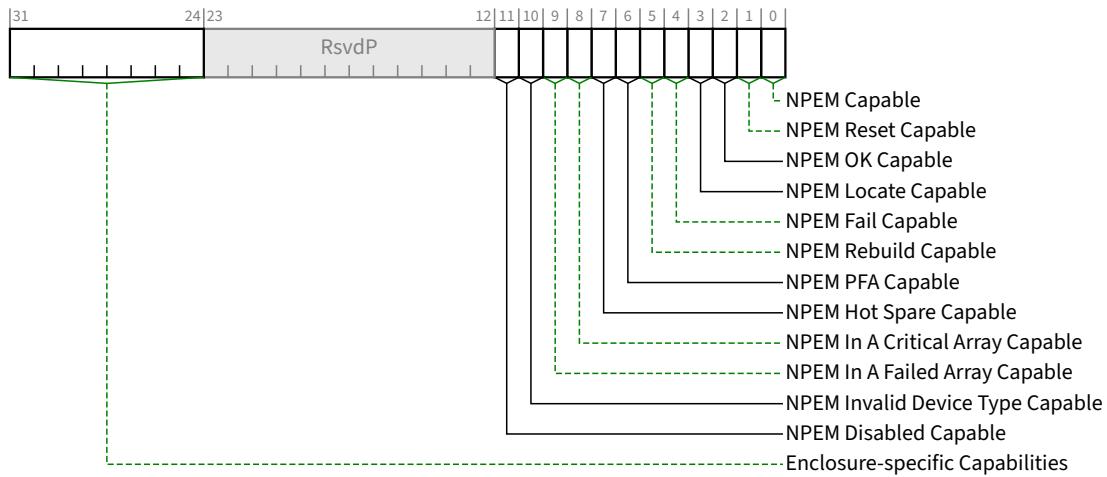


Figure 7-340 NPEM Capability Register §

Table 7-299 NPEM Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | NPEM Capable - When Set, this bit indicates that the enclosure has NPEM functionality. | HwInit |
| 1 | NPEM Reset Capable - A value of 1b indicates support for the optional NPEM Reset mechanism described in § Section 6.28 . This capability is independently optional. | HwInit |
| 2 | NPEM OK Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM OK state. This bit must be Set if NPEM Capable is also Set. | HwInit |
| 3 | NPEM Locate Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Locate state. This bit must be Set if NPEM Capable is also Set. | HwInit |
| 4 | NPEM Fail Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Fail state. This bit must be Set if NPEM Capable is also Set. | HwInit |
| 5 | NPEM Rebuild Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Rebuild state. This bit must be Set if NPEM Capable is also Set. | HwInit |
| 6 | NPEM PFA Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM PFA state. This capability is independently optional. | HwInit |
| 7 | NPEM Hot Spare Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Hot Spare state. This capability is independently optional. | HwInit |
| 8 | NPEM In A Critical Array Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM In A Critical Array state. This capability is independently optional. | HwInit |
| 9 | NPEM In A Failed Array Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM In A Failed Array state. This capability is independently optional. | HwInit |
| 10 | NPEM Invalid Device Type Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM_Invalid_Device_Type state. This capability is independently optional. | HwInit |

| Bit Location | Register Description | Attributes |
|--------------|--|---------------|
| 11 | NPEM Disabled Capable - When Set, this bit indicates that enclosure has the ability to indicate the NPEM_Disabled state. This capability is independently optional. | <u>HwInit</u> |
| 31:24 | Enclosure-specific Capabilities - The definition of enclosure-specific bits is outside the scope of this specification. | <u>HwInit</u> |

7.9.19.3 NPEM Control Register (Offset 08h) §

The NPEM Control Register contains an overall NPEM Enable bit and a bit map of states that software controls.

Use of Enclosure-specific bits is outside the scope of this specification.

All writes to this register, including writes that do not change the register value, are NPEM commands and should eventually result in a command completion indication in the NPEM Status Register .

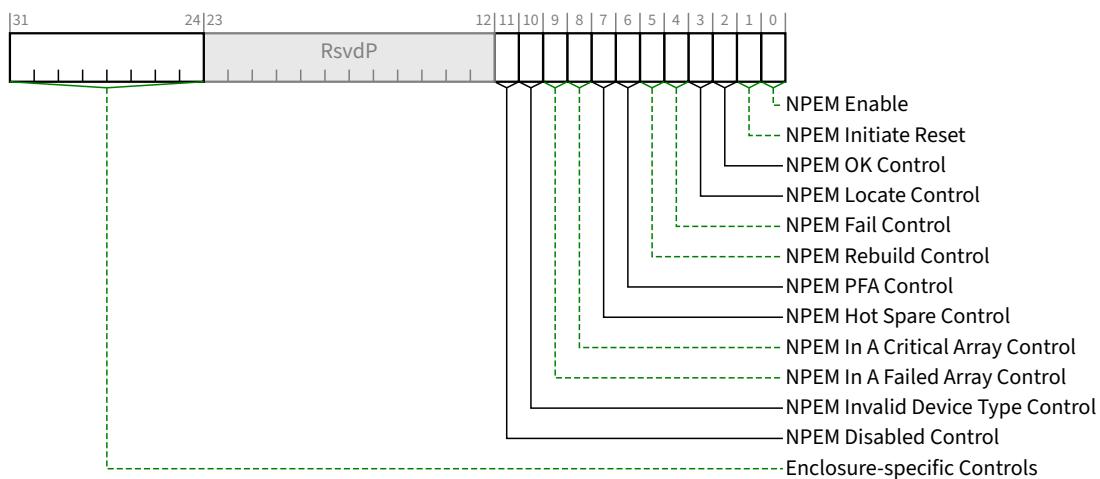


Figure 7-341 NPEM Control Register §

Table 7-300 NPEM Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| 0 | NPEM Enable - When Set, this bit enables the NPEM capability. When Clear, this bit disables the NPEM capability. Default value of this bit is 0b. When enabled, this capability operates as defined in this specification. When disabled, the other bits in this capability have no effect and any associated indications are outside the scope of this specification. | <u>RW</u> |
| 1 | NPEM Initiate Reset - If NPEM Reset Capable bit is 1b, then a write of 1b to this bit initiates NPEM Reset. If NPEM Reset Capable bit is 0b, then this bit is permitted to be read-only with a value of 0b. The value read by software from this bit must always be 0b. | <u>RW / RO</u> |
| 2 | NPEM OK Control - When Set, this bit specifies that the NPEM OK indication be turned ON. When Clear, this bit specifies that the NPEM OK indication be turned OFF. | <u>RW / RO</u> |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | If NPEM OK Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | |
| 3 | NPEM Locate Control - When Set, this bit specifies that the NPEM Locate indication be turned ON. When Clear, this bit specifies that the NPEM Locate indication be turned OFF. If NPEM Locate Capable bit in the NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 4 | NPEM Fail Control - When Set, this bit specifies that the NPEM Fail indication be turned ON. When Clear, this bit specifies that the NPEM Fail indication be turned OFF. If NPEM Fail Capable bit in the NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 5 | NPEM Rebuild Control - When Set, this bit specifies that the NPEM Rebuild indication be turned ON. When Clear, this bit specifies that the NPEM Rebuild indication be turned OFF. If NPEM Rebuild Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 6 | NPEM PFA Control - When Set, this bit specifies that the NPEM PFA indication be turned ON. When Clear, this bit specifies that the NPEM PFA indication be turned OFF. If NPEM PFA Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 7 | NPEM Hot Spare Control - When Set, this bit specifies that the NPEM Hot Spare indication be turned ON. When Clear, this bit specifies that the NPEM Hot Spare indication be turned OFF. If NPEM Hot Spare Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 8 | NPEM In A Critical Array Control - When Set, this bit specifies that the NPEM In A Critical Array indication be turned ON. When Clear, this bit specifies that the NPEM In A Critical Array indication be turned OFF. If NPEM In A Critical Array Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 9 | NPEM In A Failed Array Control - When Set, this bit specifies that the NPEM In A Failed Array indication be turned ON. When Clear, this bit specifies that the NPEM In A Failed Array indication be turned OFF. If NPEM In A Failed Array Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b | RW / RO |
| 10 | NPEM Invalid Device Type Control - When Set, this bit specifies that the NPEM Invalid <ins>Invalid</ins> Device Type indication be turned ON. When Clear, this bit specifies that the NPEM Invalid Device Type indication be turned OFF. If NPEM Invalid Device Type Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b. | RW / RO |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | Default value of this bit is 0b | |
| 11 | <p>NPEM Disabled Control - When Set, this bit specifies that the NPEM Disabled indication be turned ON. When Clear, this bit specifies that the NPEM Disabled indication be turned OFF.</p> <p>If NPEM Disabled Capable bit in NPEM Capability Register is 0b, this bit is permitted to be read-only with a value of 0b.</p> <p>Default value of this bit is 0b</p> | RW / RO |
| 31:24 | <p>Enclosure-specific Controls - The definition of enclosure-specific bits is outside the scope of this specification. Enclosure-specific software is permitted to change the value of this field. Other software must preserve the existing value when writing this register.</p> <p>Default value of this field is 00h</p> | RW / RO |

7.9.19.4 NPEM Status Register (Offset 0Ch) §

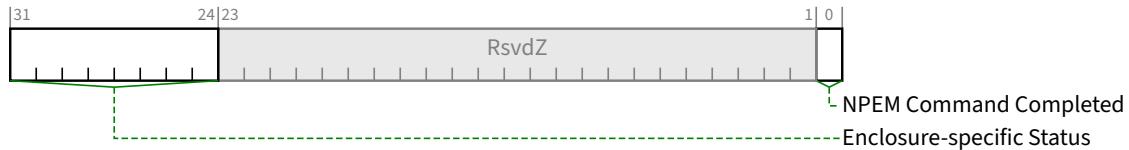


Figure 7-342 NPEM Status Register §

Table 7-301 NPEM Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|-------------------|
| 0 | <p>NPEM Command Completed - This bit is Set when an NPEM command has completed, and the NPEM controller is ready to accept a subsequent command.</p> <p>This bit is permitted to be hardwired to 1b if the enclosure is able to accept writes that update any portion of the NPEM Control register without any delay between successive writes.</p> <p>Default value of this bit is 0b.</p> <p>Software must wait for an NPEM command to complete before issuing the next NPEM command. However, if this bit is not set within 1 second limit on command execution, software is permitted to repeat the NPEM command or issue the next NPEM command. If software issues a write before the Port has completed processing of the previous command and before the 1 second time limit has expired, the Port is permitted to either accept or discard the write. Such a write is considered a programming error, and could result in a discrepancy between the NPEM Control Register and the enclosure element state. To recover from such a programming error and return the enclosure to a consistent state, software must issue a write to the NPEM Control Register which conforms to the NPEM command completion rules.</p> | RW1C / RO |
| 31:24 | <p>Enclosure-specific Status - The definition of enclosure specific bits is outside the scope of this specification. Enclosure specific software is permitted to write non-zero values to this field. Other software must write 00h to this field.</p> <p>The default value of this field is enclosure-specific.</p> <p>This field is permitted to be hardwired to 00h.</p> | RsvdZ / RO / RW1C |

7.9.20 Alternate Protocol Extended Capability §

The Alternate Protocol Extended Capability structure is optional in components that implement Alternate Protocol Negotiation. It is only permitted in:

- A Function associated with a Downstream Port.
- Function 0 (and only Function 0) of a Device associated with an Upstream Port.

§ Figure 7-343 details allocation of register fields in the Alternate Protocol Extended Capability structure.

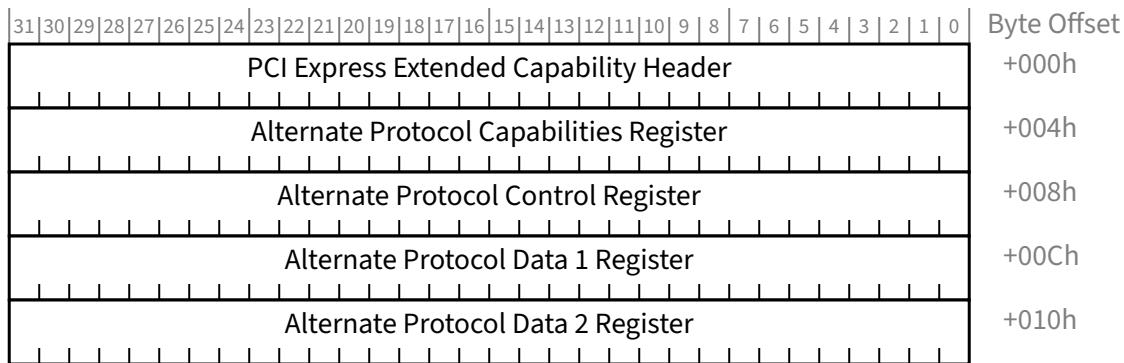


Figure 7-343 Alternate Protocol Extended Capability §

7.9.20.1 Alternate Protocol Extended Capability Header (Offset 00h) §

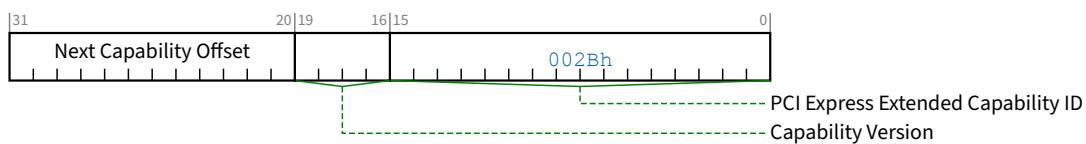


Figure 7-344 Alternate Protocol Extended Capability Header §

Table 7-302 Alternate Protocol Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Alternate Protocol Capability is 002Bh . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:20 | <p>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.</p> <p>For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.</p> | RO |

7.9.20.2 Alternate Protocol Capabilities Register (Offset 04h) [§](#)

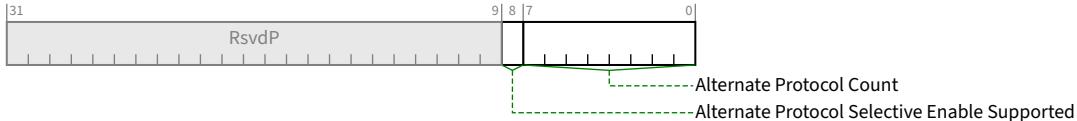


Figure 7-345 Alternate Protocol Capabilities Register [§](#)

Table 7-303 Alternate Protocol Capabilities Register [§](#)

| Bit Location | Register Description | Attributes |
|--------------|---|-------------|
| 7:0 | <p>Alternate Protocol Count - Indicates the number of Alternate Protocols or protocols that support Training Set Messages on one or more Lanes of this Link.</p> <p>The value of this field must be greater than or equal to 0.</p> | HwInit |
| 8 | <p>Alternate Protocol Selective Enable Supported - If Set, the Alternate Protocol Selective Enable Mask Register is present. If Clear, the Alternate Protocol Selective Enable Mask Register is not present and Alternate Protocol Negotiation is controlled solely by the Alternate Protocol Negotiation Global Enable bit.</p> <p>In Upstream Ports, this bit is hardwired to 0b.</p> <p>In Downstream Ports, this bit is HwInit with an implementation specific default value.</p> | RO / HwInit |

7.9.20.3 Alternate Protocol Control Register (Offset 08h) [§](#)

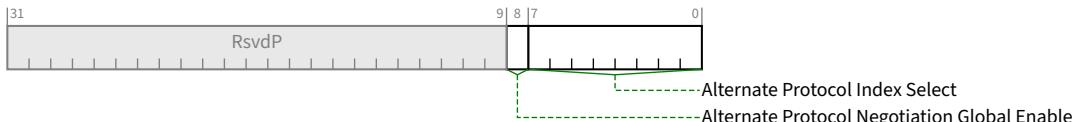


Figure 7-346 Alternate Protocol Control Register [§](#)

Table 7-304 Alternate Protocol Control Register [§](#)

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 7:0 | <p>Alternate Protocol Index Select - This field determines which Lane and which Alternate Protocol of that Lane is visible in Alternate Protocol Data 1 Register and Alternate Protocol Data 2 Register .</p> | RW |

| Bit Location | Register Description | Attributes |
|--------------|--|---|
| | <p>The default value of this field is 00h. Unused bits in this field are permitted to be hardwired to 0b.</p> <p>If <u>Alternate Protocol Count</u> is 01h, this field is permitted to be hardwired to 00h.</p> <p>Behavior is undefined if this field is greater than <u>Alternate Protocol Count</u>.</p> <p>Specific Alternate Protocol Index Select values are permitted to be disabled without renumbering other protocol index values. Disabled entries return an Alternate Protocol Vendor ID of FFFFh.</p> | |
| 8 | <p>Alternate Protocol Negotiation Global Enable - When this bit is Set, Alternate Protocol Negotiation is enabled for this Link. When this bit is Clear, Alternate Protocol Negotiation is disabled for this Link.</p> <p>This bit is <u>RW</u> for Downstream Ports. It is <u>HwInit</u> for Upstream Ports.</p> <p>Default is 0b.</p> | <u>RW / HwInit</u> (see description) |

7.9.20.4 Alternate Protocol Data 1 Register (Offset 0Ch) §

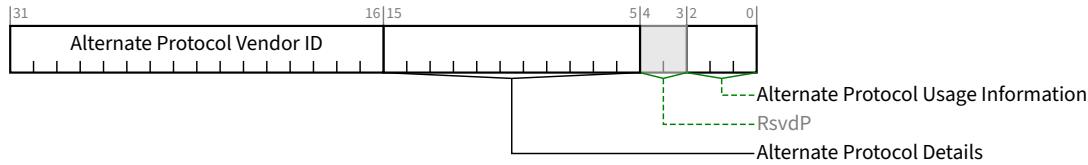


Figure 7-347 Alternate Protocol Data 1 Register §

Table 7-305 Alternate Protocol Data 1 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>Alternate Protocol Usage Information - This field contains the <u>Modified TS Usage</u> associated alternate protocol associated with the <u>Alternate Protocol Index Select</u> value.</p> <p>If <u>Alternate Protocol Vendor ID</u> is FFFFh, the value of this field is undefined.</p> | <u>RO</u> |
| 15:5 | <p>Alternate Protocol Details - This field contains the <u>Alternate Protocol Details</u> associated alternate protocol associated with the <u>Alternate Protocol Index Select</u> value.</p> <p>If <u>Alternate Protocol Vendor ID</u> is FFFFh, the value of this field is undefined.</p> | <u>RO</u> |
| 31:16 | <p>Alternate Protocol Vendor ID - This field contains the Vendor ID associated alternate protocol associated with the <u>Alternate Protocol Index Select</u> value.</p> <p>Bits 7:0 of this field contain bits 7:0 of Vendor ID (Symbol 10).</p> <p>Bits 15:8 of this field contain bits 15:8 of Vendor ID (Symbol 11).</p> <p>If <u>Alternate Protocol Index Select</u> is greater than or equal to <u>Alternate Protocol Count</u>, this field contains FFFFh.</p> <p>If <u>Alternate Protocol Index Select</u> is associated with a disabled alternate protocol, this field contains FFFFh.</p> | <u>RO</u> |

7.9.20.5 Alternate Protocol Data 2 Register (Offset 10h) §

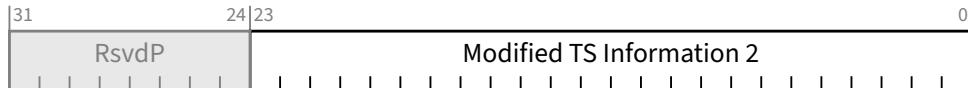


Figure 7-348 Alternate Protocol Data 2 Register §

Table 7-306 Alternate Protocol Data 2 Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 23:0 | <p>Modified TS Information 2 - This field contains the value for symbols 12 N↓throught↓ <ins>N↑throught↑</ins> 14 for the alternate protocol associated with the <u>Alternate Protocol Index Select</u> value.</p> <p>If <u>Alternate Protocol Vendor ID</u> is FFFFh, the value of this field is undefined.</p> <p>Bits 7:0 contain the value of Symbol 12.</p> <p>Bits 16:8 contain the value of Symbol 13.</p> <p>Bits 23:16 contain the value of Symbol 14.</p> | RO |

7.9.20.6 Alternate Protocol Selective Enable Mask Register (Offset 14h) §

This register is present if Alternate Protocol Selective Enable Supported is Set.

This register consists of a bit mask of size Alternate Protocol Count bits. Each bit corresponds to a valid value of Alternate Protocol Index Select. This register is an integral number of DWORDS in size.

When Alternate Protocol Negotiation Global Enable is Set, a particular bit in this register is Set, and the corresponding Alternate Protocol is not disabled (see Alternate Protocol Index Select), the next Alternate Protocol negotiation is permitted to consider using that Alternate Protocol. When a particular bit in this register is Clear, the next Alternate Protocol negotiation is not permitted to consider using the corresponding Alternate Protocol.

Changes to this field will affect the next Alternate Protocol negotiation and have no effect on current operation of the Link (regardless of current protocol).

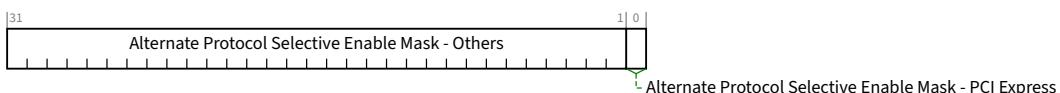


Figure 7-349 Alternate Protocol Selective Enable Mask Register §

Table 7-307 Alternate Protocol Selective Enable Mask Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Alternate Protocol Selective Enable Mask - PCI Express - The PCI Express Protocol is always index 00h. The default value of this bit is 1b (i.e., PCI Express is always enabled by default).</p> | RWS |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | This bit is permitted to be hardwired to 1b. | |
| 31:1 | <p>Alternate Protocol Selective Enable Mask - Others - Other bits in this register represent protocols other than PCI Express. The default values of these “other” bits is implementation specific.</p> <p>The width of this field is shown here as 32 bits. The actual width depends on <u>Alternate Protocol Count</u>.</p> <p>Bits in this field corresponding to disabled Alternate Protocol Index values are permitted to be hardwired to 0b.</p> <p>Bits in this field corresponding to <u>Alternate Protocol Index Select</u> values above <u>Alternate Protocol Count</u> are permitted to be hardwired to 0b.</p> | RWS |

7.9.21 Conventional PCI Advanced Features Capability (AF) §

This capability is optional. It is permitted only in Conventional PCI Functions that are integrated into a Root Complex. A Function may contain at most one instance of this capability.

§ Figure 7-350 shows the layout of this capability.

Note: Due to document production limitations, this figure shows an 8 byte capability while the actual capability is only 6 bytes long. Bytes 6 and 7 in the figure are not part of the capability.

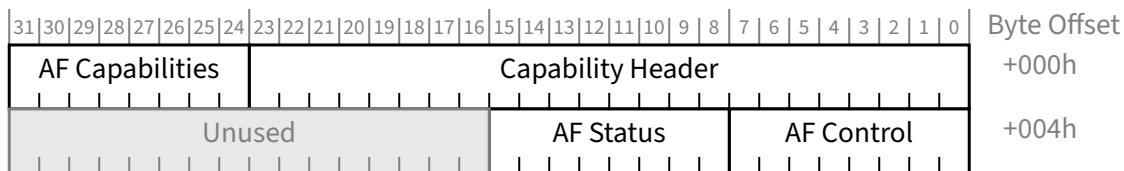


Figure 7-350 Conventional PCI Advanced Features Capability (AF) §

7.9.21.1 Advanced Features Capability Header (Offset 00h) §

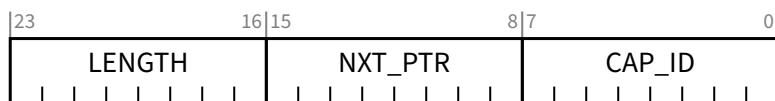


Figure 7-351 Advanced Features Capability Header §

Table 7-308 Advanced Features Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | CAP_ID - The value of 13h in this field identifies the Function as being AF capable. | RO |
| 15:8 | NXT_PTR - Pointer to the next item in the capabilities list. Must be 00h for the final item in the list. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 23:16 | LENGTH - AF Structure Length (Bytes). Shall return a value of 06h. | RO |

7.9.21.2 AF Capabilities Register (Offset 03h) §

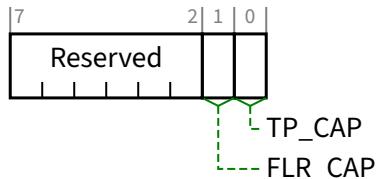


Figure 7-352 AF Capabilities Register §

Table 7-309 AF Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | TP_CAP - Set to indicate support for the Transactions Pending (TP) bit. TP_CAP must be Set if FLR_CAP is Set. | HwInit |
| 1 | FLR_CAP - Set to indicate support for Function Level Reset (INITIATE_FLR) . | HwInit |
| 7:2 | Reserved Reserved - Shall be implemented as read only returning a value of 000 0000b. | RO |

7.9.21.3 Conventional PCI Advanced Features Control Register (Offset 04h) §



Figure 7-353 Conventional PCI Advanced Features Control Register §

Table 7-310 Conventional PCI Advanced Features Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | Function Level Reset (INITIATE_FLR) - A write of 1b initiates a Function Level Reset (FLR). Registers and state information that do not apply to Conventional PCI are exempt from the FLR requirements in this specification (see § Section 6.6.2). The value read by software from this bit shall always be 0b. | RW |
| 7:1 | Reserved Reserved - Shall be implemented as read only returning a value of 000 0000b. | RO |

7.9.21.4 AF Status Register (Offset 05h) §

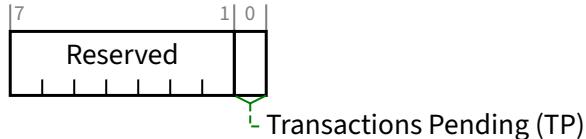


Figure 7-354 AF Status Register §

Table 7-311 AF Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Transactions Pending (TP) - A value of 1b indicates that the Function ↓↓has issued ↑↑is expecting one or more ↓↓non-posted transactions which have not been completed, including non-posted transactions that a target has terminated with Retry. ↑↑Completions. A value 0b indicates that ↓↓all non-posted transactions have been completed. ↑↑no Completions are expected.</p> <p style="text-align: right;">Errata: Base 6.3 B817△◀</p> | RO |
| 7:1 | Reserved Reserved - Shall be implemented as read only returning a value of 000 0000b. | RO |

7.9.22 SFI Extended Capability §

The SFI (System Firmware Intermediary) Extended Capability is an optional capability that provides system firmware with enhanced control over primarily hot-plug mechanisms, and enables system firmware to operate as an intermediary between certain events and the operating system (see § Section 6.7.4). This capability may be implemented by a Root Port or a Switch Downstream Port. It is not applicable to any other Device/Port type.

If a Downstream Port implements the ~~↓↓SFI Extended Capability,~~ ~~↑↑SFI Extended Capability,~~ that Port must support ERR_COR Subclass capability, and indicate so by Setting the ERR_COR Subclass Capable bit in the Device Capabilities Register . See ~~↓↓see~~ § Section 7.5.3.3 .

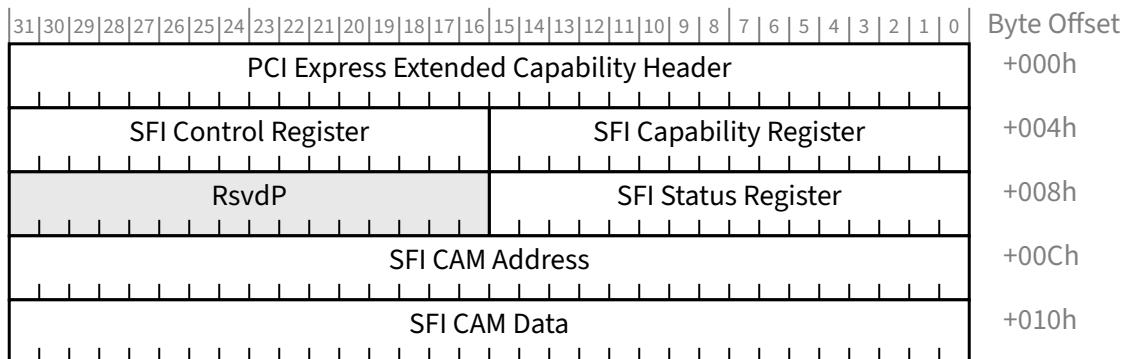


Figure 7-355 SFI Extended Capability §

7.9.22.1 SFI Extended Capability Header (Offset 00h) §

§ Figure 7-356 and § Table 7-312 detail allocation of fields in the Extended Capability header.

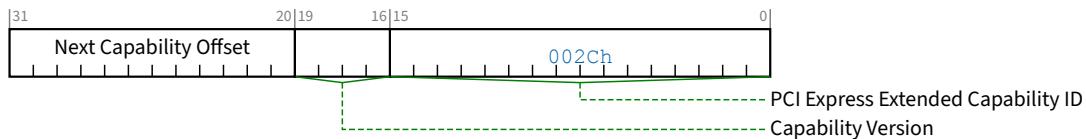


Figure 7-356 SFI Extended Capability Header §

Table 7-312 SFI Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the SFI Extended Capability is 002Ch . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.9.22.2 SFI Capability Register (Offset 04h) §

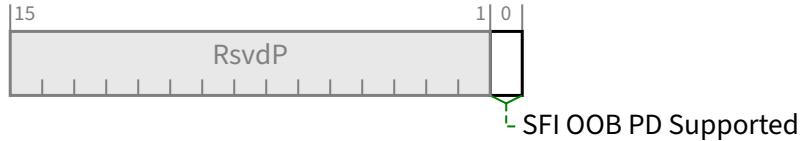


Figure 7-357 SFI Capability Register §

Table 7-313 SFI Capability Register §

| Bit Location | Register Description | Attributes |
|----------------------------|---|------------|
| 0 | <p>SFI OOB PD Supported - When Set, this bit indicates that this slot supports reporting the out-of-band presence detect state. If this Downstream Port has no implemented slot (as indicated by the Slot Implemented bit in the PCI Express Capabilities Register), then the value of this bit must be 0b.</p> <p>↑↑Note that similar functionality is architected for the SCap2 OOB PD Supported field, which is intended for use when the SFI Extended Capability is either not implemented or is hidden when accessed by Configuration Read Requests or Configuration Write Requests (e.g., initiated by in-band management system software). See the SFI Hidden In-Band bit for details.↑</p> | ↑↑HwInit↑ |
| 1↑↑1↑ ECN: Base 6.3 eSFI△↔ | <p>↑↑Enhanced SFI Supported – If this bit Set, this SFI Extended Capability supports a defined set of enhancements collectively referred to as Enhanced SFI or eSFI; otherwise, this SFI Extended Capability does not support eSFI.↑</p> | ↑↑HwInit↑ |
| 1↑↑2↑ ECN: Base 6.3 eSFI△↔ | <p>↑↑SFI CAM Unsupported – If this bit is Set, the SFI CAM is not supported; otherwise, the SFI CAM is supported.↑</p> <p>↑↑See § Section 6.7.4.3 for details.↑</p> | ↑↑HwInit↑ |
| 1↑↑3↑ ECN: Base 6.3 eSFI△↔ | <p>↑↑SFI Hidden In-Band – If this bit is Set, this SFI Extended Capability is hidden when accessed using Configuration Read Requests or Configuration Write Requests (e.g., initiated by in-band management system software) but is not hidden when accessed using PCIe-MI ; otherwise, this SFI Extended Capability is not hidden when accessed using Configuration Read Requests, Configuration Write Requests, or PCIe-MI .↑</p> <p>↑↑If eSFI is not supported, this bit must be Clear.↑</p> <p>↑↑See § Section 6.7.4.7 for details.↑</p> | HwInit |

7.9.22.3 SFI Control Register (Offset 06h) §

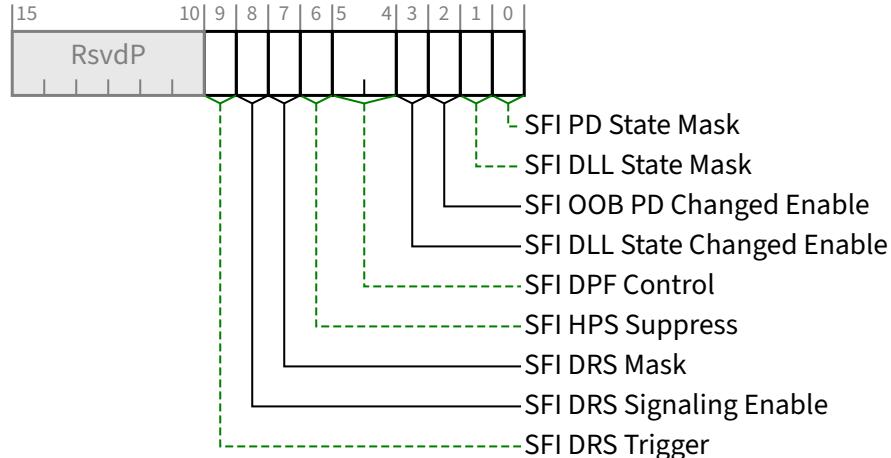


Figure 7-358 SFI Control Register §

Table 7-314 SFI Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>SFI PD State Mask - When Set, this bit masks the Presence Detect State bit in the Slot Status Register, making its value 0b, regardless of the actual presence detect state. Otherwise, its value indicates the actual ↑↑state.↓ ECN: Base 6.3 eSFI△↔ ↑↑statestate, unless otherwise specified (e.g., the Presence Detect State bit is masked by the SFI Quarantine bit being Set)..↑</p> <p>If the value of the Presence Detect State bit changes when the SFI PD State Mask bit value changes, this must cause a Presence Detect Changed event (see § Section 6.7.3).</p> <p>Default value of this bit is 0b.</p> | RW |
| 1 | <p>SFI DLL State Mask - When Set, this bit masks the Data Link Layer Link Active bit in the Link Status Register, making its value 0b, regardless of the actual Data Link Layer state. Otherwise, its value indicates the actual state.</p> <p>If the value of the Data Link Layer Link Active State bit changes when the SFI DLL State Mask bit value changes, this must cause a Data Link Layer State Changed event (see § Section 6.7.3).</p> <p>Default value of this bit is 0b.</p> | RW |
| 2 | <p>SFI OOB PD Changed Enable - When Set, this bit enables sending an ERR_COR Message for the SFI OOB PD Changed event. See § Section 6.7.4.1 for other necessary conditions.</p> <p>This bit must be RW if the SFI OOB PD Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. If the SFI OOB PD Supported bit is Clear and software Sets this bit, the behavior is undefined.</p> <p>Default value of this bit is 0b.</p> | RW / RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes | | | | | | | | |
|--------------|--|------------|----------|------------|---|------------|---|------------|---|----|
| 3 | <p>SFI DLL State Changed Enable - When Set, this bit enables sending an ERR_COR Message for the SFI DLL State Changed event. See § Section 6.7.4.1 for other necessary conditions.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | |
| 5:4 | <p>SFI DPF Control - This field controls the level of Downstream Port Filtering (DPF) enabled on the Downstream Port, governing which Request TLPs targeting Downstream Components get filtered; that is, Port are filtered, or blocked. A filtered Request TLP must be handled as if the Link is in DL_Down. If an Unsupported Request and a filtered Completion TLP must be silently discarded (following update of flow control credits for upstream flowing TLPs). See § Section 6.7.4.2.</p> <p>Defined encodings are:</p> <table> <tr> <td>00b</td><td>Disabled</td></tr> <tr> <td>01b</td><td>Filter all Downstream Request TLPs except Configuration Request TLPs generated by this Port's SFI CAM mechanism</td></tr> <tr> <td>10b</td><td>Filter only Downstream Configuration Request TLPs that are not generated by this Port's SFI CAM mechanism</td></tr> <tr> <td>11b</td><td>Reserved. Filter all Upstream/Downstream TLPs except (a) MCTP VDMs and (b) Configuration Request TLPs generated by this Port's SFI CAM mechanism when accessed using PCIe-MI and the Completion TLPs for those Configuration Request TLPs. If eSFI is not supported, this encoding must not be supported and hardware behavior is undefined if this encoding is programmed.</td></tr> </table> <p>Default If eSFI is supported, the default value of this field is implementation-specific; otherwise, the default value of this field is 00b.</p> | 00b | Disabled | 01b | Filter all Downstream Request TLPs except Configuration Request TLPs generated by this Port's SFI CAM mechanism | 10b | Filter only Downstream Configuration Request TLPs that are not generated by this Port's SFI CAM mechanism | 11b | Reserved. Filter all Upstream/Downstream TLPs except (a) MCTP VDMs and (b) Configuration Request TLPs generated by this Port's SFI CAM mechanism when accessed using PCIe-MI and the Completion TLPs for those Configuration Request TLPs. If eSFI is not supported, this encoding must not be supported and hardware behavior is undefined if this encoding is programmed. | RW |
| 00b | Disabled | | | | | | | | | |
| 01b | Filter all Downstream Request TLPs except Configuration Request TLPs generated by this Port's SFI CAM mechanism | | | | | | | | | |
| 10b | Filter only Downstream Configuration Request TLPs that are not generated by this Port's SFI CAM mechanism | | | | | | | | | |
| 11b | Reserved. Filter all Upstream/Downstream TLPs except (a) MCTP VDMs and (b) Configuration Request TLPs generated by this Port's SFI CAM mechanism when accessed using PCIe-MI and the Completion TLPs for those Configuration Request TLPs. If eSFI is not supported, this encoding must not be supported and hardware behavior is undefined if this encoding is programmed. | | | | | | | | | |
| 6 | <p>SFI HPS Suppress - When Set, this bit forces the Hot-Plug Surprise (HPS) bit in the Slot Capabilities Register to be Clear and disables associated Hot-Plug Surprise functionality. See § Section 6.7.4.4.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | |
| 7 | <p>SFI DRS Mask - When Set, this bit masks the DRS Message Received bit in the Link Status 2 Register, making its value 0b, regardless of the actual DRS Message Received state. Otherwise, its value indicates the actual state.</p> <p>If the value of the DRS Message Received bit changes from Clear to Set when the SFI DRS Mask bit is Cleared, this must trigger any notification enabled by the DRS Signaling Control field in the Link Control Register (see § Section 7.5.3.7).</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | |
| 8 | <p>SFI DRS Signaling Enable - When Set, this bit enables sending an ERR_COR Message for the SFI DRS Received event. See § Section 6.7.4.1 for other necessary conditions.</p> <p>Default value of this bit is 0b.</p> | RW | | | | | | | | |
| 9 | <p>SFI DRS Trigger - If the SFI DRS Mask bit is Clear, when software writes a 1b to this bit, the Downstream Port must behave as if a DRS Message was received. Otherwise, software writing a 1b to this bit has no effect.</p> | RW | | | | | | | | |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|----------------------------------|--|------------------|
| ↑↑11:10↑ ECN: Base 6.3 eSFI△↔ | <p>It is permitted to write 1b to this bit while simultaneously writing updated values to other fields in this register, notably the SFI DRS Mask bit. For this case, the SFI DRS Trigger semantics are based on the updated value of the SFI DRS Mask bit.</p> <p>This bit always returns 0b when read.</p> <p>↑↑SFI Quarantine Mode – This field determines if/when a Port is quarantined. When a Port transitions from not quarantined to quarantined, the SFI Quarantine bit must be Set which causes Functions below the Port to be (a) hidden when accessed using Configuration Read Requests and Configuration Write Requests (e.g., initiated by in-band management system software) and (b) blocked from injecting traffic into the platform over the Link. See § Section 6.7.4.6 for details.↑</p> <p>↑↑Defined encodings are:↑</p> <ul style="list-style-type: none"> ↑↑00b↑ The Port does not transition to quarantined due to any condition↑ ↑↑01b↑ The Port transitions to quarantined upon the SFI PD State bit or the SFI OOB PD State bit transitioning from Set to Clear (e.g., due to the hot removal of an adapter)↑ ↑↑10b↑ The Port transitions to quarantined upon the SFI PD State bit or the SFI OOB PD State bit transitioning from Set to Clear or due to DL_Down (e.g., due the hot removal of an adapter or DL_Down due to reasons other than hot removal such as the operating system disabling the Link)↑ ↑↑11b↑ The Port transitions to quarantined upon this field being modified to a value of 11b due to a Conventional Reset if the default value of this field is 11b or due to a write to this field↑ <p>↑↑Default value of this field is implementation specific.↑</p> <p>↑↑If eSFI is supported, this field must be RO when accessed using a Configuration Read Request or Configuration Write Request (e.g., initiated by in-band management system software) and must be RW when accessed using PCIe-MI.↑</p> <p>↑↑If eSFI is not supported, this field must be read-only and hardwired to 00b.↑</p> | ↑↑RW / RO↑ |
| ↑↑12↑ ECN: Base 6.3 eSFI△↔ | <p>↑↑SFI RO In-Band – This field controls whether Configuration Read Requests and Configuration Write Requests (e.g., initiated by in-band management system software) to this SFI Extended Capability structure are forced to be read-only. See § Section 6.7.4.8 for details.↑</p> <p>↑↑Default value of this bit is implementation specific.↑</p> <p>↑↑If eSFI is not supported, this field must be read-only and hardwired to 0b.↑</p> | ↑↑HwInit /RO↑ |

7.9.22.4 SFI Status Register (Offset 08h) §

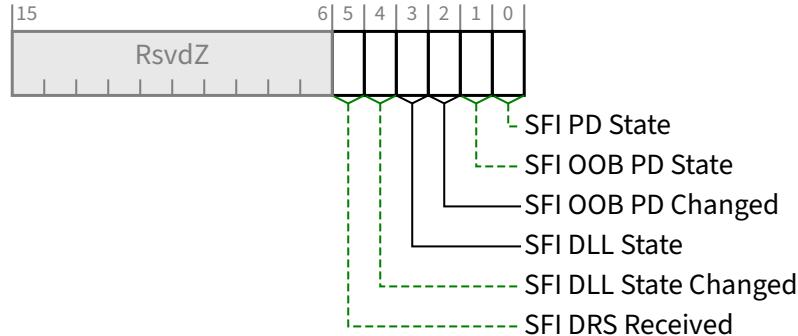


Figure 7-359 SFI Status Register §

Table 7-315 SFI Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|---|
| 0 | SFI PD State - This bit always indicates the actual presence detect state associated with the Presence Detect State bit in the Slot Status Register, even when the value of that bit is being masked by the SFI PD State Mask bit. | RO |
| 1 | SFI OOB PD State - This bit indicates the out-of-band presence detect state, independent of the in-band presence detect state. This bit must be implemented if the SFI OOB PD Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. | RO |
| 2 | SFI OOB PD Changed - This bit is Set when the value reported in the SFI OOB PD State bit is changed. Default value of this bit is 0b.1 | RW1C |
| 3 | SFI DLL State - This bit always indicates the actual link state associated with the Data Link Layer Link Active bit in the Link Status Register, even when the value of that bit is being masked by the SFI DLL State Mask bit. | RO |
| 4 | SFI DLL State Changed - This bit is Set when the value reported in the SFI DLL State bit is changed. Default value of this bit is 0b.1 | RW1C |
| 5 | SFI DRS Received - This bit always indicates the actual state associated with the DRS Message Received bit in the Link Status 2 Register, even when the value of that bit is being masked by the SFI DRS Mask bit. Clearing the SFI DRS Received bit (by writing a 1b to it) must also cause the actual state associated with the DRS Message Received bit to be Cleared. Default value of this bit is 0b.1 | RW1C |

Errata: Base 6.3
B828△◀▶

Errata: Base 6.3
B828△◀▶

Errata: Base 6.3 B828△◀▶

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|---------------------------|--|-----------------|
| ↑↓6↑ ECN: Base 6.3 eSFI△↔ | <p>↑↓SFI Quarantine – This bit is Set when the Port transitions to quarantined due to the conditions described in the SFI Quarantine Mode field. Clearing this bit results in the Port being taken out of quarantine. See § Section 6.7.4.6 for details.↑</p> <p>↑↓If eSFI is supported, this field must be RO when accessed using a Configuration Read Request or Configuration Write Request (e.g., initiated by in-band management system software) and must be RW1C when accessed using PCIe-MI.↑</p> <p>↑↓If eSFI is not supported, this bit must be read-only and hardwired to 0b.↑</p> <p>↑↓If the SFI Quarantine Mode field default is 11b, the default value of this bit is 1b; otherwise, the default value of this bit is 0b.↑</p> | RW1C ↑↓/ RO↑ |

7.9.22.5 SFI CAM Address Register (Offset 0Ch) §



Figure 7-360 SFI CAM Address Register §

Table 7-316 SFI CAM Address Register §

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| 27:0 | <p>SFI CAM Address - This field specifies the target Bus, Device, and Function Numbers, along with the Extended Register Number and Register Number, in the format specified by § Table 7-1 .</p> <p>↑↓If the SFI CAM Unsupported bit is Set, this field must still be writeable unless otherwise specified (e.g., the SFI RO In-Band bit is Set and this field is accessed by a Configuration Write Request) and must be readable. The value of this field must still be used by any architected error handling due to accesses to the SFI CAM Data field.↑</p> | RW ↑↓/ RO↑ |

7.9.22.6 SFI CAM Data Register (Offset 10h) §



Figure 7-361 SFI CAM Data Register §

Table 7-317 SFI CAM Data Register §

| Bit Location | Register Description | Attributes |
|--------------|---|---------------|
| 31:0 | <p>SFI CAM Data - ↑↓When this field is ↑↓read,↑↓Clear, the following rules apply.↑</p> | RW ↑↓/ RO↑ |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|---|----------------------|------------|
| <ul style="list-style-type: none"> If the SFI RO In-Band bit is Clear, a Configuration Read Request (e.g., initiated by in-band management system software) must cause the SFI CAM to generate and transmit a Configuration Read Request on the Link below this Port. A read to this field using PCIe-MI must cause the SFI CAM to generate and transmit a Configuration Read Request on the Link below this Port. If the SFI RO In-Band bit is written, Clear, a Configuration Write Request must cause the SFI CAM to generate and transmit a Configuration Write Request on the Link below this Port. A write to this field using PCIe-MI must cause the SFI CAM to generate and transmit a Configuration Write Request on the Link below this Port. In both cases, the target of the Configuration Request is determined by the value of the SFI CAM Address Register . See § Section 6.7.4.3 . <p>If the SFI CAM Unsupported bit is Set, the following rules apply.</p> <ul style="list-style-type: none"> This field must still be RW. If the SFI RO In-Band bit is Clear, a Configuration Read Request or a Configuration Write Request to this field must be handled as an Unsupported Request. A read or write to this field using PCIe-MI must be handled as an Unsupported Request. <p>If the SFI RO In-Band bit is Set, the following rules apply.</p> <ul style="list-style-type: none"> If this field is accessed by a Configuration Read Request, a value of 0h must be returned without generating Configuration Read Request on the Link below this Port. If this field is accessed by a Configuration Write Request, this field must not be altered, and a Configuration Write Request must not be generated on the Link below this Port. | | |

7.9.23 Subsystem ID and Subsystem Vendor ID Capability §

The Subsystem ID and Subsystem Vendor ID Capability is an optional capability used to uniquely identify the add-in card or subsystem where the PCI device resides. It provides a mechanism for add-in card vendors to distinguish their add-in cards from one another even though the add-in cards may have the same PCI bridge on them (and, therefore, the same Vendor ID and Device ID). The format of the capability is shown in § Figure 7-362 . The fields are described in § Table 7-318 and § Table 7-319 .

This capability is only permitted in Functions with Type 1 Configuration Space Headers .

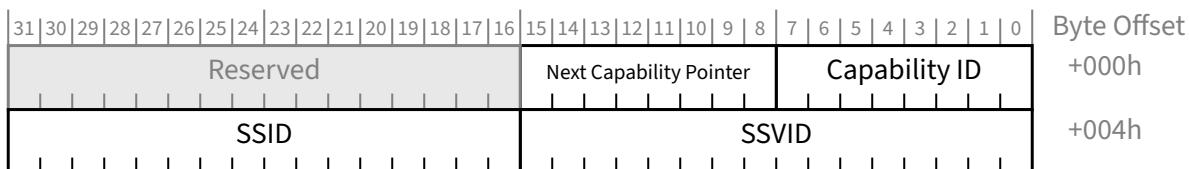


Figure 7-362 Subsystem ID and Subsystem Vendor ID Capability §

7.9.23.1 Subsystem ID and Subsystem Vendor ID Capability Header (Offset 00h) §

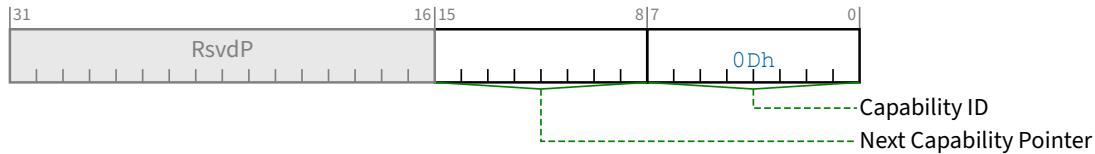


Figure 7-363 Subsystem ID and Subsystem Vendor ID Capability Header §

Table 7-318 Subsystem ID and Subsystem Vendor ID Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | Capability ID - Indicates the PCI Express Capability structure. This field must return a Capability ID of 0Dh indicating that this is a Subsystem ID and Subsystem Vendor ID Capability structure. | RO |
| 15:8 | Next Capability Pointer - This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities. | RO |

7.9.23.2 Subsystem ID and Subsystem Vendor ID Capability Data (Offset 04h) §

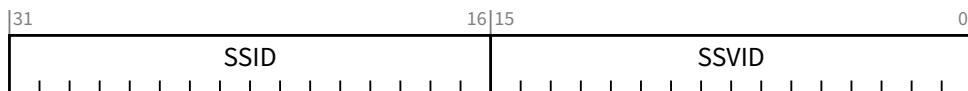


Figure 7-364 Subsystem ID and Subsystem Vendor ID Capability Data §

Table 7-319 Subsystem ID and Subsystem Vendor ID Capability Data §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | SSVID - The SSVID identifies the manufacturer of the add-in card or subsystem. The SSVID is assigned by PCI-SIG to insure uniqueness (the Vendor ID is used as the SSVID also). This field is read-only. | HwInit |
| 31:16 | SSID - The SSID identifies the particular add-in card or subsystem and is assigned by the vendor. This field is read-only. | HwInit |

7.9.24 Data Object Exchange Extended Capability §

The Data Object Exchange (DOE) Extended Capability is an optional Extended Capability for discovering and controlling a mechanism for the exchange of data objects (see § Section 6.30). It is permitted for a Function to implement more than one instance of this Extended Capability.

§ Figure 7-365 illustrates the Data Object Exchange Extended Capability structure.

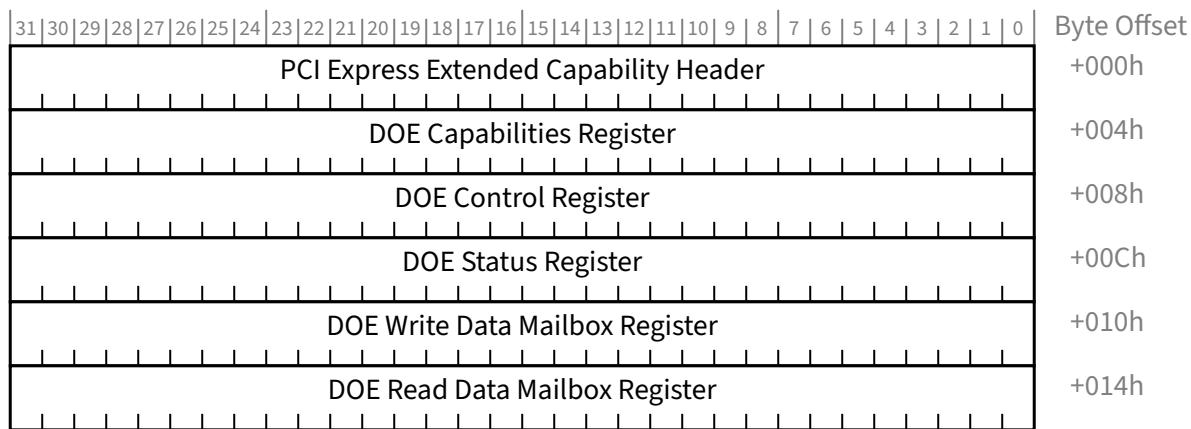


Figure 7-365 Data Object Exchange Extended Capability §

7.9.24.1 DOE Extended Capability Header (Offset 00h) §

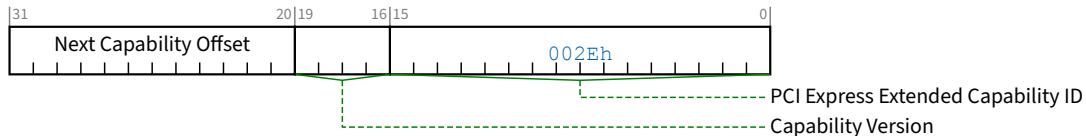


Figure 7-366 DOE Extended Capability Header §

Table 7-320 DOE Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | <p>PCI Express Extended Capability ID – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.</p> <p>The Extended Capability ID for the Data Object Exchange Extended Capability is 002Eh .</p> | RO |
| 19:16 | <p>Capability Version – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.</p> <p>Must be 2h for this version of the specification.</p> <p>New implementations compliant to the older version of this specification must indicate Capability Version 1h.</p> | RO |
| 31:20 | <p>Next Capability Offset – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.</p> | RO |

7.9.24.2 DOE Capabilities Register (Offset 04h) §

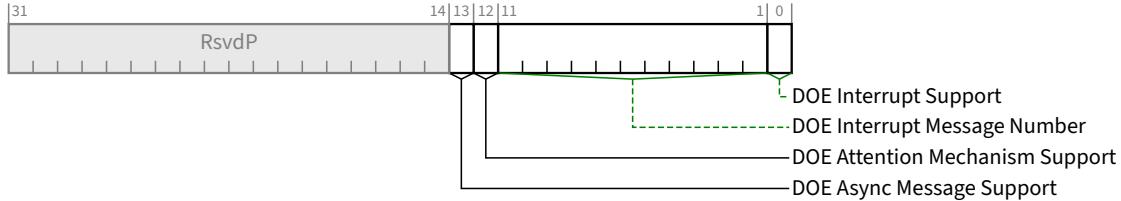


Figure 7-367 DOE Capabilities Register §

Table 7-321 DOE Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | DOE Interrupt Support – When Set, this bit indicates DOE support of software notification of DOE events using MSI/MSI-X. | HwInit |
| 11:1 | <p>DOE Interrupt Message Number – When the DOE Interrupt Support bit is Set, this field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with DOE.</p> <p>For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control Register for MSI.</p> <p>For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. For a given MSI-X implementation, the entry must remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.</p> <p>When the DOE Interrupt Support bit is Clear the value in this field is undefined.</p> | RO |
| 12 | DOE Attention Mechanism Support – This bit, when Set, indicates the DOE instance supports the optional DOE Attention mechanism. | HwInit |
| 13 | DOE Async Message Support – This bit, when Set, indicates the DOE instance supports the optional DOE Async Message mechanism. | HwInit |

7.9.24.3 DOE Control Register (Offset 08h) §

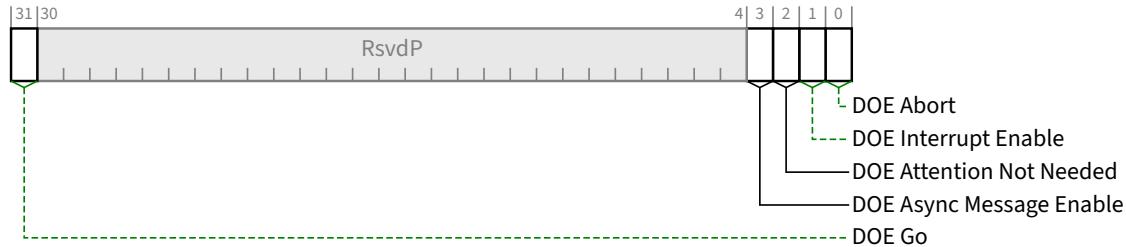


Figure 7-368 DOE Control Register §

Table 7-322 DOE Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|----------------------|
| 0 | <p>DOE Abort – A write of 1b to this bit must cause all data object transfer operations associated with this DOE instance to be aborted.</p> <p>Reads from this bit must always return 0b.</p> | RW (see description) |
| 1 | <p>DOE Interrupt Enable – When this bit is Set, the <u>DOE Interrupt Support</u> bit is Set, and MSI/MSI-X is enabled, the DOE instance must issue an MSI/MSI-X interrupt as defined in § Section 6.30.3 .</p> <p>When <u>DOE Interrupt Support</u> is Clear, this bit is permitted to be Reserved.</p> <p>Default value of this bit is 0b.</p> | RW / RsvdP |
| 2 | <p>DOE Attention Not Needed – When <u>DOE Attention Mechanism Support</u> is Set, this bit when Set enables the DOE instance to enter and stay in a state where it is not immediately available for use. When this bit is Clear the DOE instance must remain in a responsive state.</p> <p>When <u>DOE Attention Mechanism Support</u> is Clear, this bit is permitted to be Reserved.</p> <p>Default value of this bit is 0b.</p> | RW / RsvdP |
| 3 | <p>DOE Async Message Enable – If <u>DOE Async Message Support</u> is Set, this bit, when Set, enables the use of the DOE Async Message mechanism.</p> <p>When <u>DOE Async Message Support</u> is Clear, this bit is permitted to be Reserved.</p> <p>Default value of this bit is 0b.</p> | RW / RsvdP |
| 31 | <p>DOE Go – A write of 1b to this bit indicates to the DOE instance that it can start consuming the data object transferred through the <u>DOE Write Data Mailbox Register</u> .</p> <p>Behavior is undefined if the <u>DOE Go</u> bit is Set before the entire data object has been written to the <u>DOE Write Data Mailbox Register</u> .</p> <p>Behavior is undefined if the <u>DOE Go</u> bit is written with 1b when the <u>DOE Busy</u> bit is Set.</p> <p>Reads from this bit must always return 0b.</p> | RW (see description) |

7.9.24.4 DOE Status Register (Offset 0Ch) §

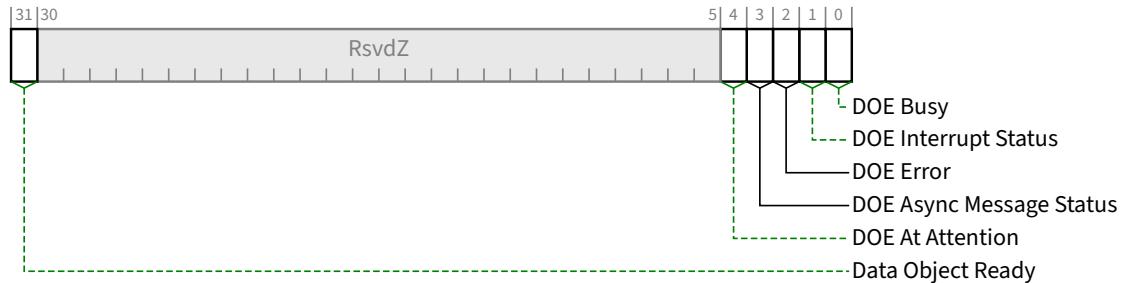


Figure 7-369 DOE Status Register §

Table 7-323 DOE Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|--------------|
| 0 | <p>DOE Busy — When Set, this bit indicates the DOE instance is temporarily unable to receive a new data object through the <u>DOE Write Data Mailbox Register</u>.</p> <p>The DOE instance must Set this bit when processing a received data object, and Clear this bit when it is able to receive a new data object.</p> <p>The DOE instance must Set this bit following an abort or reset if, as a result of the abort/reset, it is temporarily unable to receive a data object, and then must Clear this bit when it is able to receive a new data object.</p> | RO |
| 1 | <p>DOE Interrupt Status – If <u>DOE Interrupt Support</u> is Set, then this bit must be Set when an interrupt-triggering event occurs.</p> <p>If <u>DOE Interrupt Support</u> is Clear, this bit is Reserved.</p> <p>Default value of this bit is 0b.</p> | RW1C / RsvdZ |
| 2 | <p>DOE Error – This bit, when Set, indicates that there has been an internal error associated with a data object received, or that a data object has been received for which the DOE instance is unable to provide a response.</p> <p>The DOE instance must Clear this bit, if it is not already Clear, when 1b is written to the <u>DOE Abort bit</u> in the <u>DOE Control Register</u>. Writing 1b to the <u>DOE Abort bit</u> is the only mechanism for software to Clear this bit.</p> <p>The transition of this bit from Clear to Set is an interrupt triggering event.</p> <p>Default value of this bit is 0b.</p> | RO |
| 3 | <p>DOE Async Message Status – If <u>DOE Async Message Support</u> is Set, this bit, when Set, indicates the DOE instance has one or more asynchronous messages to transfer.</p> <p>The transition of this bit from Clear to Set is an interrupt triggering event.</p> <p>If <u>DOE Async Message Support</u> is Clear, this bit is Reserved.</p> <p>Default value of this bit is 0b.</p> | RO / RsvdZ |
| 4 | <p>DOE At Attention – When <u>DOE Attention Mechanism Support</u> is Set, this bit, when Set, indicates the DOE interface is presently in a state of readiness.</p> <p>The transition of this bit from Clear to Set is an interrupt triggering event.</p> | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | When DOE Attention Mechanism Support is Clear, this bit is Reserved. | |
| 31 | <p>Data Object Ready – When Set, this bit indicates the DOE instance has a data object available to be read by system firmware/software.</p> <p>If there is no additional data object ready for transfer, the DOE instance must clear this bit after the entire data object has been transferred, as indicated by software writing to the <u>DOE Read Data Mailbox Register</u> after reading the final DW of the data object.</p> <p>The DOE instance must clear this bit, if not already clear, upon a write of 1b to the <u>DOE Abort</u> bit in the <u>DOE Control Register</u>.</p> <p>The transition of this bit from Clear to Set is an interrupt triggering event.</p> <p>Default value of this bit is 0b.</p> | RO |

7.9.24.5 DOE Write Data Mailbox Register (Offset 10h) §

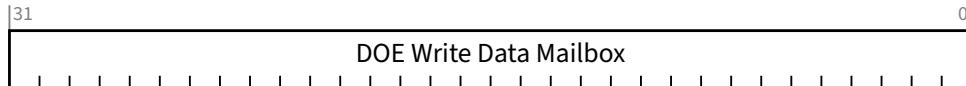


Figure 7-370 DOE Write Data Mailbox Register §

Table 7-324 DOE Write Data Mailbox Register §

| Bit Location | Register Description | Attributes |
|--------------|---|----------------------|
| 31:0 | <p>DOE Write Data Mailbox – The DOE instance receives data objects via writes to this register.</p> <p>A successfully completed write to this register adds one DW to the incoming data object.</p> <p>Setting the <u>DOE Go</u> bit in the <u>DOE Control Register</u> indicates to the DOE Instance that the final DW of the data object has been written to this register.</p> <p>Reads of this register must return all 0's.</p> | RW (see description) |

7.9.24.6 DOE Read Data Mailbox Register (Offset 14h) §

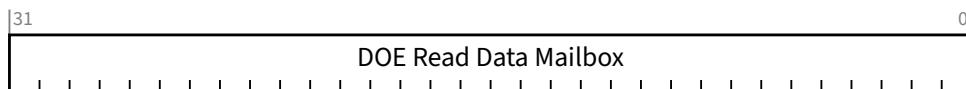


Figure 7-371 DOE Read Data Mailbox Register §

Table 7-325 DOE Read Data Mailbox Register §

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------|
| 31:0 | <p>DOE Read Data Mailbox – If the <u>Data Object Ready</u> bit is Set, a read of this register returns the current DW of the data object.</p> <p>A write of any value to this register indicates a successful transfer of the current data object DW, and the DOE instance must return the next DW in the data object upon the next read of this register as long as the <u>Data Object Ready</u> bit remains Set.</p> <p>It is permitted for multiple data objects to be read from this register back-to-back. When this scenario occurs, the <u>Data Object Ready</u> bit will remain Set until this register is written after the final DW is read.</p> <p>A write of any value to this register when the <u>Data Object Ready</u> bit is Clear must have no effect.</p> <p>The value read from this register when the <u>Data Object Ready</u> bit is Clear must be 0000 0000h.</p> | RW (see description) |

7.9.25 Shadow Functions Extended Capability §

Unimplemented Functions possess Transaction ID resources by virtue of their Bus/Device/Function Number space, and therefore associated Requester ID space, and associated Tags, even though there is no Function implemented there to use them. The **Shadow Functions Extended Capability** is an optional capability that permits a Requester to use the Transaction ID resources of another otherwise unimplemented Function to generate more outstanding Requests than it would otherwise be able to using only the Transaction ID resources of the Function it is associated with. The Requester generates some of its Requests via the Function it is associated with and generates other Requests via the Shadow Function. If the Requester exceeds the Transaction ID resources of a single Function, it is permitted to implement this capability and split its Transaction ID space across that Function and additional **Shadow Functions** defined by this capability.

A Requester implementing a Shadow Function uses the characteristics and attributes of the Function containing this capability. Requests made via the associated Function will use the associated Function's BDF to populate the Requester ID. Requests made via the Shadow Function will use the BDF calculated from the value in the Shadow Function Number field of the corresponding Shadow Function Instance register entry to populate the Requester ID. Other characteristics and attributes of the Shadow Function are taken from the associated Function's Configuration Space.

The Shadow Function Number field in the Shadow Function Instance register entry for each Shadow Function is used to calculate the value of the Bus/Device/Function number (Bus/Function number for ARI devices) (BDF) for that Shadow Function. That BDF space assigned to the Shadow Function must be available, that is it corresponds to an otherwise unimplemented Function.

Additional requirements for implementing **Shadow Functions** are:

- Any access to the Configuration Space region of the BDF associated with the Shadow Function, without errors that would have different behavior, must be responded to with a Completion with UR status.
- For non-ARI Devices, the Shadow Function must reside in the same Device as the Function it is shadowing. ARI must be supported if the Shadow Function Number is greater than 7.
- A Function is permitted to have more than one Shadow Function.
- A Function is permitted to have at most one instance of this capability.
- This capability is permitted to be implemented in any Function capable of operating as a Requester.
- For VFs, the **Shadow Functions** must be assigned in a manner that accommodates the VF Discovery algorithm (see § Section 9.2.1.2).
- Requesters are permitted to generate Posted Requests that are not Message Signaled Interrupt (MSI/MSI-X) Requests using the Transaction ID space of a Shadow Function.

- Requesters are not permitted to generate Message Signaled Interrupt (MSI/MSI-X) Requests using the Transaction ID space of a Shadow Function.
- Functions utilizing **↓↑Shadow Functions↓↑Shadow Functions** must be aware that accesses utilizing the Shadow Function's Transaction ID resources appear to the rest of the system with the same semantics as if the access was from any independent Function and deal with those implications.
- The software for the Translation Agent is responsible for maintaining the integrity of address translation resources. Behavior is undefined if address translation resources are not updated before the Shadow Function's Requester makes a Request.
- Translation Requests issued by a Shadow Function are cached in the ATC associated with the main Function. When enabled, Functions are permitted to use translations across the “main” and **↓↑Shadow Functions↓↑Shadow Functions** regardless of which Function issued the associated Translation Request. See § Section 10.2.
- The software for handling Page Request Messages is responsible for coordinating usage across Shadow Functions. See § Section 10.4.1 and § Section 10.5.2.5.
- Behavior is undefined if software enabling FPB configures a Shadow Function to use the same Requester ID as another Function.
- For a Multi-Function Device that supports ACS P2P Egress Control , any enabled **↓↑Shadow Functions↓↑Shadow Functions** must be taken into account when configuring the Egress Control Vector to allow P2P traffic between the Requester and its Shadow Functions, and other Functions in the Device.

IMPLEMENTATION NOTE: SHADOW FUNCTION NUMBER PROGRAMMING §

The value programmed into the Shadow Function Number field should place the Shadow Function on the same Bus Number as the Function declaring it. Otherwise, ACS Source Validation might not operate appropriately, Completions targeting the Shadow Function might not be routed correctly, or other misbehaviors might occur.

Multiple **↓↑Shadow Functions↓↑Shadow Functions** for a Function are permitted to be assigned by this Capability. The Number of **↓↑Shadow Functions↓↑Shadow Functions** field in the **↓↑Shadow Functions↓↑Shadow Functions** Capability register defines the number of **↓↑Shadow Functions↓↑Shadow Functions** assigned and the number of Shadow Function Instance register entries in the Capability and therefore the length of the Capability structure.

§ Figure 7-372 shows the **↓↑Shadow Functions↓↑Shadow Functions** Extended Capability structure.

```
↓↓{"debug": false, "preClass": "hide", "cellwidth": 15, "width": 224, "isMemoryBlock": true, "defaultUnused": "R", "fields": [ {"lsbyte": 0, "msbyte": 3, "msbit": 7, "name": "PCI Express Extended Capability Header", "attr": "ro"}, { "lsbyte": 4, "msbyte": 7, "msbit": 7, "name": "Shadow Functions Capability Register", "attr": "ro"}, { "lsbyte": 8, "msbyte": 11, "msbit": 7, "name": "Shadow Functions Control Register", "attr": "rw"}, { "lsbyte": 12, "msbyte": 15, "msbit": 7, "name": "Shadow Functions Instance Register 0", "attr": "rw"}, { "lsbyte": 16, "msbyte": 19, "msbit": 7, "name": "Shadow Functions Instance Register 1", "attr": "rw"}, { "lsbyte": 20, "msbyte": 23, "msbit": 7, "name": "...", "attr": "rw"}, { "lsbyte": 24, "msbyte": 27, "msbit": 7, "name": "Shadow Functions Instance Register N", "attr": "rw"}]}↓
```

Base 6.4 vs Base 6.3

N = the value of the Number of Shadow Functions field.

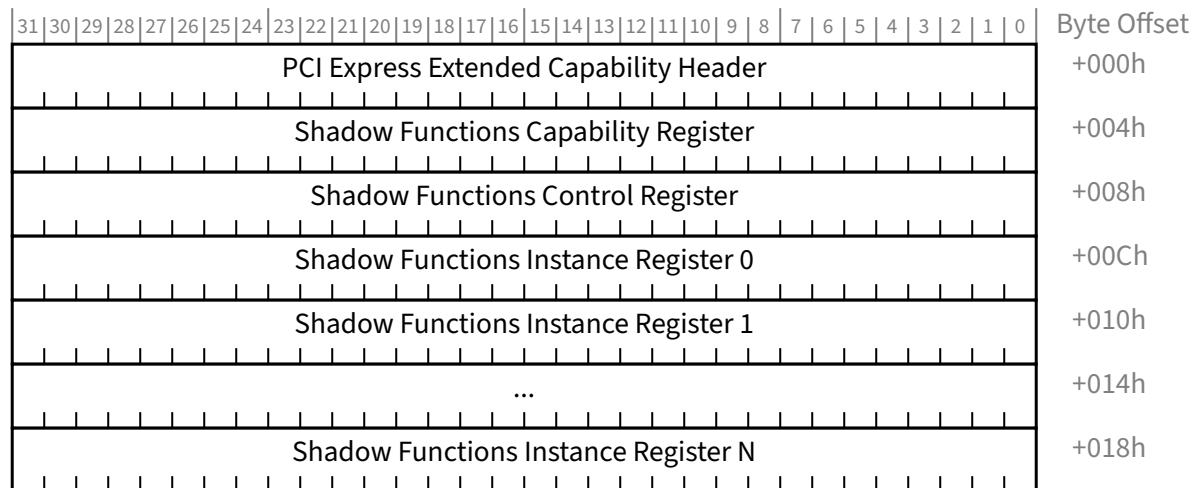


Figure 7-372 Shadow Functions Extended Capability Structure §

7.9.25.1 Shadow Functions Extended Capability Header (Offset 00h) §

§ Figure 7-373 details allocation of the register fields in the Shadow Functions Extended Capability Header ; § Table 7-326 provides the respective bit definitions.

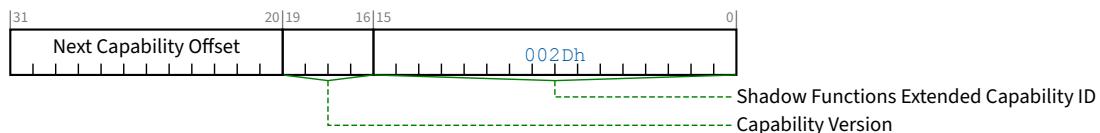


Figure 7-373 Editorial Shadow Functions Extended Capability Header Editorial Shadow Functions Extended Capability Header §

Table 7-326 ~~Shadow Functions Extended Capability Header~~ Shadow Functions Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | Shadow Functions Extended Capability ID - Indicates the Shadow Functions Extended Capability structure. This field must return a Capability ID of 002Dh indicating that this is a Shadow Functions Extended Capability structure. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

7.9.25.2 Shadow Functions Capability Register (Offset 04h) §

§ Figure 7-374 details the allocation of register bits of the [Shadow Functions Capability Register](#); § Table 7-327 provides the respective bit definitions.

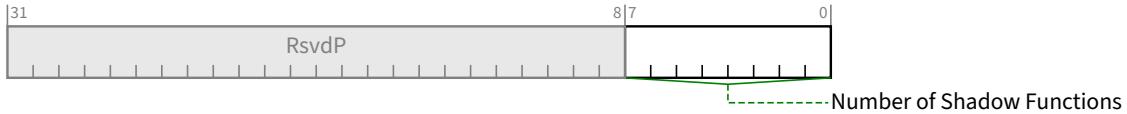


Figure 7-374 Editorial [Shadow Functions Capability Register](#) Editorial [Shadow Functions Capability Register](#) §

Table 7-327 [Shadow Functions Capability Register](#) [Shadow Functions Capability Register](#) §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | Number of Shadow Functions – This is one less than the number of Shadow Functions implemented by this Function. This defines the number of Shadow Function Instance register entries that are in the Capability, and therefore the length of the Capability structure. The default value for this field is 00h. | HwInit |

7.9.25.3 Shadow Functions Control Register (Offset 08h) §

§ Figure 7-375 details the allocation of register bits of the [Shadow Functions Control register](#); § Table 7-328 provides the respective bit definitions.

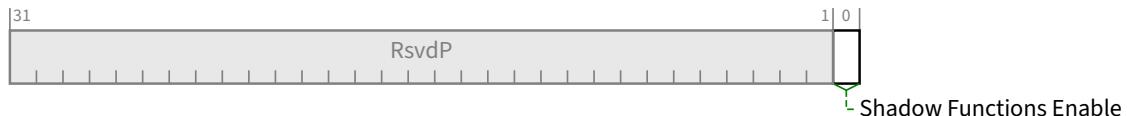


Figure 7-375 Editorial [Shadow Functions Control Register](#) Editorial [Shadow Functions Control Register](#) §

Table 7-328 [Shadow Functions Control Register](#) [Shadow Functions Control Register](#) §

| Bit Location | Register Description | Attributes |
|------------------------------|---|------------|
| 0 | Shadow Functions Enable - When Set, permits the Requester to generate Requests using the Transaction ID resources of all of the Shadow Functions defined by this Capability. See § Section 7.9.25 for limitations on the type of Requests permitted. When Clear, the Requester is not permitted to generate Requests using the Transaction ID resources of any of the Shadow Functions defined by this Capability. Behavior is undefined when this bit is Set in Functions with the Phantom Functions Enabled bit Set. | RW |

| ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|------------------------------------|--|------------|
| | <p>Behavior is undefined if the value of this bit is changed while the Function has outstanding Non-Posted Requests.</p> <p>Default is 0b.</p> | |

7.9.25.4 Shadow Functions Instance Register Entry §

§ Figure 7-376 details the allocation of register bits of the ~~↓↑Shadow Functions Control register;↓↑Shadow Functions Instance Register Entry;~~ § Table 7-329 provides the respective bit definitions.

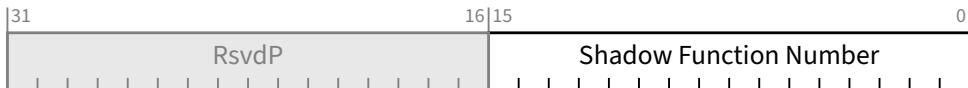


Figure 7-376 EditorialShadow Functions Instance Register Entry EditorialShadow Functions Instance Register Entry §

Table 7-329 ~~↓↑Shadow Functions Instance Register Entry;~~ ~~↑↑Shadow Functions Instance Register Entry;~~ §

| ↓↑Bit Location↓ ↑↓Bit Location↑ | Register Description | Attributes |
|------------------------------------|--|------------|
| 15:0 | Shadow Function Number - This is the Bus/Device/Function offset (Bus/Function offset for ARI Devices) of the Shadow Function. Add this value to BDF of the Function with this capability using unsigned, 16-bit arithmetic, ignoring any carry. | HwInit |

7.9.26 IDE Extended Capability §

All Ports that implement IDE must implement the IDE Extended Capability. The IDE Extended Capability must consist of the IDE Extended Capability Header , the IDE Capability Register, and the IDE Control Register, followed by zero to 8 Link IDE register blocks, followed by zero to 255 Selective IDE register blocks (see § Figure 7-377). All Ports that implement IDE must implement the IDE Extended Capability. The IDE Extended Capability must consist of the IDE Extended Capability Header , the IDE Capability Register, and the IDE Control Register, followed by zero to 8 Link IDE register blocks, followed by zero to 255 Selective IDE register blocks (see § Figure 7-377).

It is permitted to implement this extended capability in Functions associated with Downstream Ports, and in Function 0 associated with an Upstream Port. Multi-Function Devices associated with Upstream Ports, including cases where one or more Functions represent the Upstream Port of a Switch, must be implemented such that Function 0 implements this extended capability representing the Multi-Function Device as a whole.

Base 6.4 vs Base 6.3

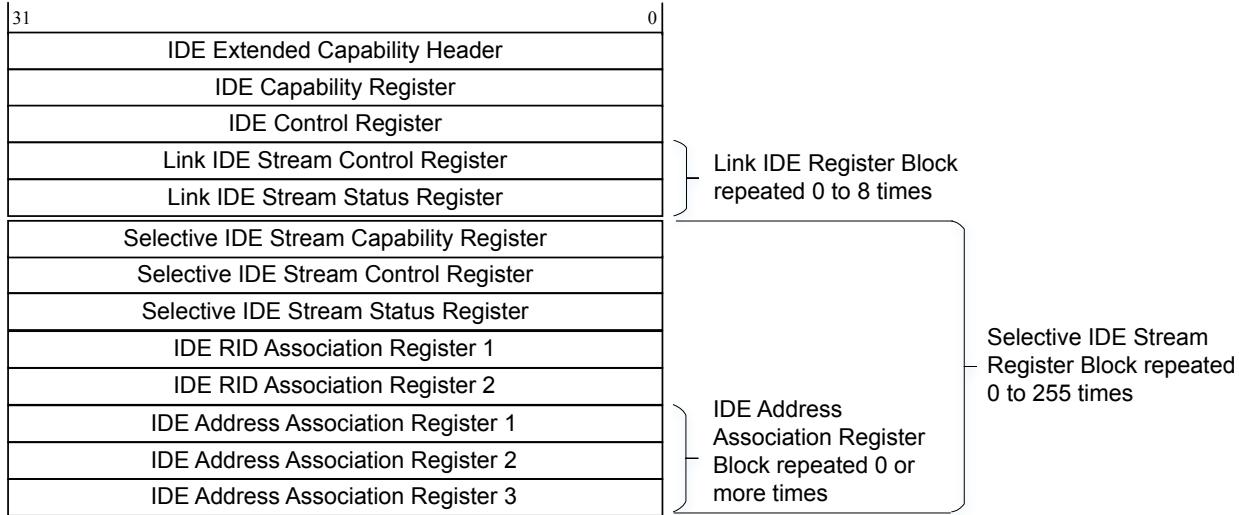


Figure 7-377 IDE Extended Capability Structure §

7.9.26.1 IDE Extended Capability Header (Offset 00h) §

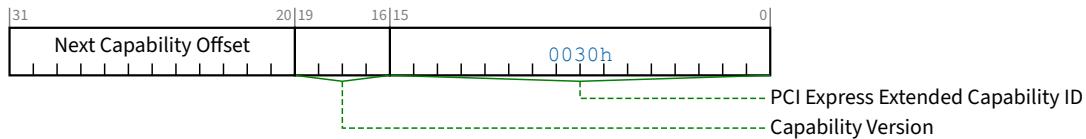


Figure 7-378 IDE Extended Capability Header §

Table 7-330 IDE Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | <p>PCI Express Extended Capability ID – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.</p> <p>The Extended Capability ID for the Integrity and Data Encryption (IDE) Exchange Extended Capability is 0030h.</p> | HwInit |
| 19:16 | <p>Capability Version – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.</p> <p>Must be 1h for this version of the specification.</p> | HwInit |
| 31:20 | Next Capability Offset – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities. | HwInit |

7.9.26.2 IDE Capability Register (Offset 04h) §

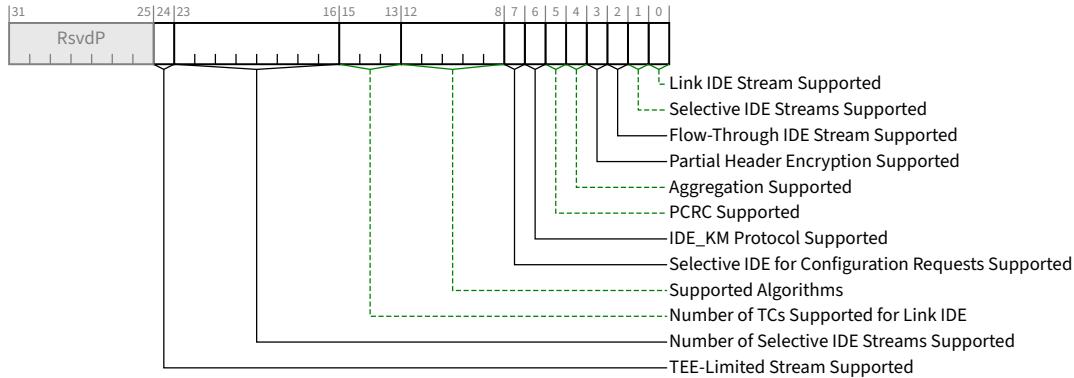


Figure 7-379 IDE Capability Register §

Table 7-331 IDE Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|----------------|
| 0 | <p>Link IDE Stream Supported – When Set, indicates that the Port supports Link IDE Streams, and that one or more Link IDE Stream Registers block(s) immediately follow the IDE Control Register, per the value in the <u>Number of TCs Supported for Link IDE</u> field.</p> <p>When Clear, there must be no Link IDE Stream Register blocks present.</p> | HwInit / RsvdP |
| 1 | <p>Selective IDE Streams Supported – When Set, indicates that the Port support Selective IDE Streams, and that one or more Selective IDE Stream Register block(s) are implemented, per the value in the <u>Number of Selective IDE Streams Supported</u> field.</p> <p>When Clear, there must be no Selective IDE Stream Register blocks present.</p> | HwInit / RsvdP |
| 2 | <p>Flow-Through IDE Stream Supported – For a Switch or Root Port, when Set indicates support for passing Selective IDE Streams to all other Switch or Root Ports.</p> <p>If this bit is Set and both <u>Link IDE Stream Supported</u> and <u>Selective IDE Streams Supported</u> are Clear, then no Link IDE register blocks or Selective IDE register blocks are required.</p> <p>Reserved for Endpoints.</p> | HwInit / RsvdP |
| 3 | <p>Partial Header Encryption Supported – If <u>Link IDE Stream Supported</u> or <u>Selective IDE Streams Supported</u> are Set, then this bit, when Set, indicates the Port supports partial header encryption.</p> <p>Undefined if <u>Link IDE Stream Supported</u> and <u>Selective IDE Streams Supported</u> are both Clear.</p> | HwInit |
| 4 | <p>Aggregation Supported – If <u>Link IDE Stream Supported</u> or <u>Selective IDE Streams Supported</u> are Set, then this bit, when Set, indicates the Port supports aggregation.</p> <p>Undefined if <u>Link IDE Stream Supported</u> and <u>Selective IDE Streams Supported</u> are both Clear.</p> | HwInit |
| 5 | <p>PCRC Supported – When Set, indicates that the Port supports the generation and checking of PCRC.</p> | HwInit |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------------------------|
| 6 | IDE_KM Protocol Supported – When Set, indicates that the Port supports the IDE_KM protocol in the responder role as defined in § Section 6.33.3 | HwInit |
| 7 | <p>Selective IDE for Configuration Requests Supported – For a Root Port, Switch Upstream Port, or Endpoint Upstream Port, if Selective IDE Streams Supported is Set, then this bit, if Set, indicates that the Port supports the ↓↑association↓ <ins>↓↑association↑</ins> of Configuration Requests with Selective IDE Streams.</p> <p>For a Switch Upstream Port, when Set, this bit indicates the Switch supports Selective IDE for Configuration Requests targeting all Functions of the Switch.</p> <p>This bit is Reserved for Switch Downstream Ports.</p> <p>If Selective IDE Streams Supported is Clear, this bit is Reserved.</p> | HwInit / RsvdP |
| 12:8 | <p>Supported Algorithms – Indicates the supported algorithms for securing IDE TLPs, encoded as:</p> <ul style="list-style-type: none"> 0 0000b AES-GCM 256 key size, 96b MAC Others Reserved | HwInit |
| 15:13 | <p>Number of TCs Supported for Link IDE – If Link IDE Stream Supported is Set, indicates the number of TCs supported for Link IDE Streams encoded as:</p> <ul style="list-style-type: none"> 000b One TC supported 001b 2 TCs supported 010b 3 TCs supported 011b 4 TCs supported 100b 5 TCs supported 101b 6 TCs supported 110b 7 TCs supported 111b 8 TCs supported <p>If Link IDE Stream Supported is Clear, this field is undefined.</p> | HwInit |
| 23:16 | <p>Number of Selective IDE Streams Supported – If Selective IDE Streams Supported is Set then this field indicates number of Selective IDE Streams Supported such that 0=1 Stream.</p> <p>A corresponding number of <u>Selective IDE Stream Register Block</u> (s) must be implemented. If <u>Link IDE Stream Supported</u> is Clear, then these blocks must immediately follow the IDE Control Register. If Link IDE Stream Supported is Set, then these blocks must immediately follow the Link IDE Stream Control and Status Registers.</p> <p>If Selective IDE Streams Supported is Clear, this field is undefined.</p> | HwInit / RsvdP |
| 24 | <p>TEE-Limited Stream Supported – When Set, indicates that the TEE-Limited Stream control mechanism is supported.</p> <p>If Selective IDE Streams Supported is Clear, this bit is Reserved.</p> | HwInit / RsvdP |
| 25 | <p>↓↑XT Supported – When Set, indicates that the Port supports sending and receiving IDE TLPs with the XT bit content.<ins>↑</ins></p> <p>↓↑Reserved if TEE-IO Supported bit in the Device Capabilities Register is Clear.<ins>↑</ins></p> | <ins>↓↑HwInit / RsvdP↑</ins> |

↓↑25↓ ECN: Base 6.3 XT△↔

7.9.26.3 IDE Control Register (Offset 08h) §

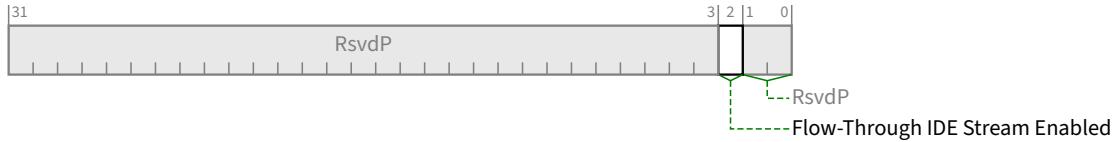


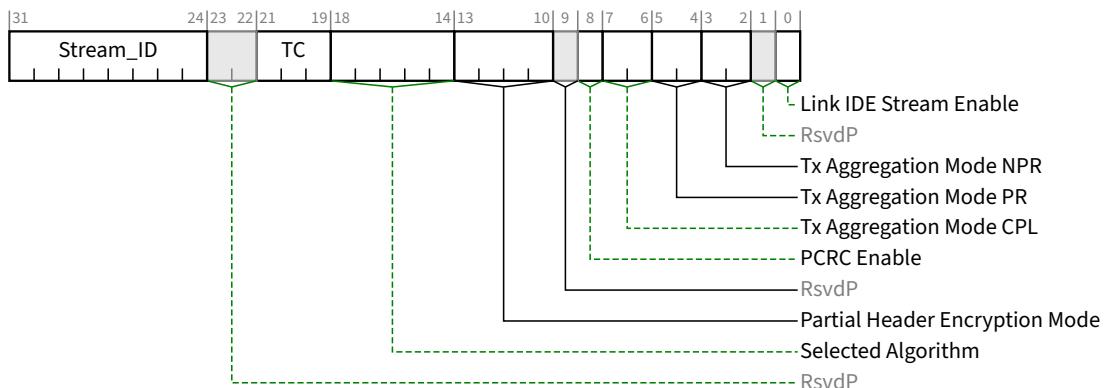
Table 7-332 IDE Control Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 2 | Flow-Through IDE Stream Enabled – For Switch Ports and Root Ports, Enables the Port for flow-through operation of TLPs associated with Selective IDE Streams. Reserved for Upstream Ports associated with Endpoints. | RW / RsvdP |

7.9.26.4 Link IDE Register Block §

A Link IDE register block must consist of one Link IDE Stream Control Register followed by one Link IDE Stream Status Register . If the Link IDE Stream Supported bit in the IDE Capability Register is Set, then this register block must be instantiated once for each Traffic Class (TC) supported as indicated in the Number of TCs Supported for Link IDE field.

7.9.26.4.1 Link IDE Stream Control Register §



Base 6.4 vs Base 6.3

Table 7-333 Link IDE Stream Control Register §

| Bit Location | Register Description | Attributes |
|--|---|--------------------------|
| 0 | <p>Link IDE Stream Enable – When Set, enables Link IDE Stream such that IDE operation will start when triggered by means of the IDE_KM protocol (see § Section 6.33.3). When Cleared, must immediately transition the Stream to Insecure.</p> <p>Software must not modify the PCRC Enable bit while this bit is Set; otherwise, the result is undefined.</p> <p>It is permitted for the default value to be 1b if and only if implementation specific means can ensure that the Link IDE Stream will default into a state where operation in the Secure state is possible, otherwise the default value must be 0b.</p> | RW |
| ↓↑1↓ ECN: Base 6.3 XT△↔ | <p>↓↑XT Enable – Controls the generation and processing of the XT bit in IDE TLPS associated with this Stream (see § Section 6.33.4).↑</p> <p>↓↑Must be configured while Link IDE Stream Enable is Clear, during which time this bit is RW. When Link IDE Stream Enable is set to 1b, the setting is sampled, and this field becomes RO with reads returning the sampled value.↑</p> <p>↓↑RsvdP if XT Supported is Clear.↑</p> <p>↓↑Default value is 0b.↑</p> | ↓↑RW / RO / RsvdP↑ |
| 3:2 | <p>Tx Aggregation Mode NPR – If Aggregation Supported is Set then this field selects the level of aggregation for Transmitted Non-Posted Requests for this Stream, encoded as:</p> <ul style="list-style-type: none"> 00b No aggregation 01b Up to 2 Non-Posted Requests 10b Up to 4 Non-Posted Requests 11b Up to 8 Non-Posted Requests <p>Reserved If Aggregation Supported is Clear.</p> <p>Default value is 00b</p> | RW / RsvdP |
| 5:4 | <p>Tx Aggregation Mode PR – If Aggregation Supported is Set then this field selects the level of aggregation for Transmitted Posted Requests for this Stream, encoded as:</p> <ul style="list-style-type: none"> 00b No aggregation 01b Up to 2 Posted Requests 10b Up to 4 Posted Requests 11b Up to 8 Posted Requests <p>Reserved If Aggregation Supported is Clear.</p> <p>Default value is 00b</p> | RW / RsvdP |
| 7:6 | <p>Tx Aggregation Mode CPL – If Aggregation Supported is Set then this field selects the level of aggregation for ↓↑Trasmitted↓ ↑↑Transmitted↑ Completions for this Stream, encoded as:</p> <ul style="list-style-type: none"> 00b No aggregation 01b Up to 2 Completions 10b Up to 4 Completions 11b Up to 8 Completions <p>Reserved If Aggregation Supported is Clear.</p> <p>Default value is 00b</p> | RW / RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|-----------------|
| 8 | <p>PCRC Enable – When Set, Transmitted IDE TLPs associated with this Stream that include P content must include PCRC, and Received TLPs must be checked for PCRC failure.</p> <p>Reserved if <u>PCRC Supported</u> is Clear.</p> <p>Default value is 0b.</p> | RW / RsvdP |
| 13:10 | <p>Partial Header Encryption Mode – Selects the mode to be used for partial header encryption of IDE TLPs for this IDE Stream. Must be programmed to the same value in both the Partner Ports. Must be configured while <u>Link IDE Stream Enable</u> is Clear. When <u>Link IDE Stream Enable</u> is Set, the setting is sampled, and this field becomes RO with reads returning the sampled value.</p> <p>0000b No partial header encryption</p> <p>0001b Address[17:2] Encrypted, and, if present, the First DW BE and Last DW BE fields</p> <p>0010b Address[25:2] Encrypted, and, if present, the First DW BE and Last DW BE fields</p> <p>0011b Address[33:2] Encrypted, and, if present, the First DW BE and Last DW BE fields</p> <p>0100b Address[41:2] Encrypted, and, if present, the First DW BE and Last DW BE fields</p> <p>Others Reserved</p> <p>If Partial Header Encryption Supported is Clear, this field is Reserved.</p> | RW / RO / RsvdP |
| 18:14 | <p>Selected Algorithm – Selects the algorithm to be used for securing IDE TLPs for this IDE Stream. Must be programmed to the same value in both the Upstream and Downstream Ports. Must be configured while <u>Link IDE Stream Enable</u> is Clear. When <u>Link IDE Stream Enable</u> is Set, the setting is sampled, and this field becomes RO with reads returning the sampled value.</p> <p>0 0000b AES-GCM 256 key size, 96b MAC</p> <p>Others Reserved</p> | RW / RO |
| 21:19 | <p>TC – System firmware/software must program this field to indicate the TC associated with this Link IDE Register block.</p> <p>Default value is 000b</p> | RW / RsvdP |
| 31:24 | <p>↓↑Stream_ID↓↑↑Stream_ID↑ – Indicates the ↓↑Stream_ID↓↑↑Stream_ID↑ associated with this Link IDE Stream. Software must program the same ↓↑Stream_ID↓↑↑Stream_ID↑ into both Ports associated with a given Link IDE Stream. Default value is 00h.</p> | RW |

7.9.26.4.2 Link IDE Stream Status Register §

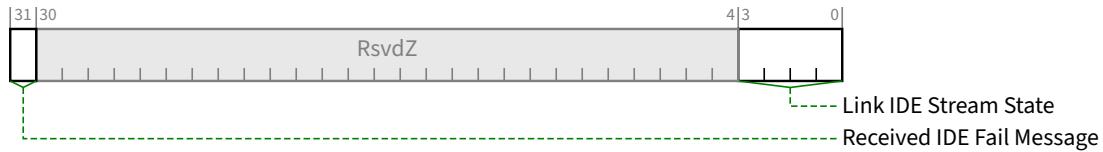


Figure 7-382 Link IDE Stream Status Register §

Table 7-334 Link IDE Stream Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | <p>Link IDE Stream State – When Link IDE Stream Enable is Set, this field indicates the state of the Port.</p> <p>Encodings:</p> <ul style="list-style-type: none"> 0000b Insecure 0010b Secure Others Reserved – Software must handle reserved values as indicating unknown state <p>When Link IDE Stream Enable is Clear, the value of this field must be 0000b.</p> | RO |
| 31 | Received IDE Fail Message – When Set, indicates that one or more IDE Fail Message(s) have been Received for this Stream. | RW1C |

7.9.26.5 Selective IDE Stream Register Block §

A Selective IDE Stream register block must consist of one Selective IDE Stream Capability Register , followed by one Selective IDE Stream Control Register , followed by one Selective IDE Stream Status Register , followed by one Selective IDE RID Association register Block , followed by zero or more Selective IDE Address Association Register Block (s) . If the Selective IDE Streams Supported bit in the IDE Capability Register is Set, then this register block must be instantiated once for each Selective IDE Stream supported as indicated in the Number of Selective IDE Streams Supported field.

7.9.26.5.1 Selective IDE Stream Capability Register §



Figure 7-383 Selective IDE Stream Capability Register §

Table 7-335 Selective IDE Stream Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | <p>Number of Address Association Register Blocks – Indicates the number of Selective IDE Address Association register blocks for this Selective IDE Stream.</p> <p>The number of Selective IDE Address Association register blocks for a given IDE Stream is hardware implementation specific, and is permitted to be any number between 0 and 15.</p> | RO |

7.9.26.5.2 Selective IDE Stream Control Register §

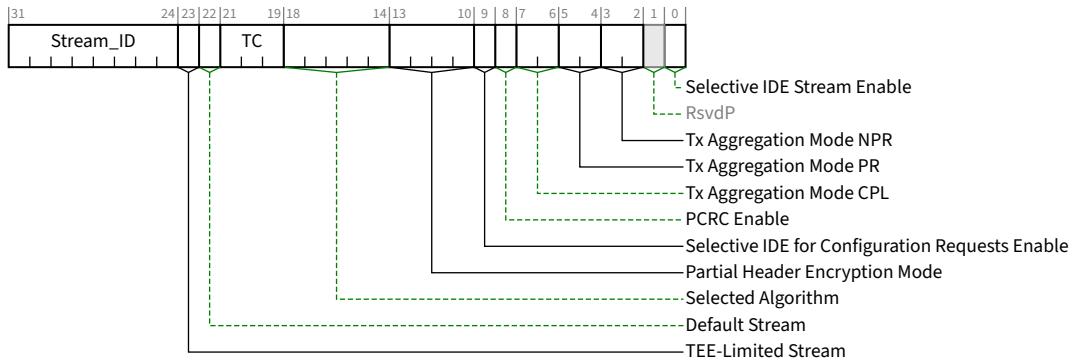


Figure 7-384 Selective IDE Stream Control Register §

Table 7-336 Selective IDE Stream Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Selective IDE Stream Enable – When Set, enables this IDE Stream such that IDE operation will start when triggered by means of the IDE_KM protocol (see § Section 6.33.3). When Cleared, must immediately transition the Stream to Insecure. Software must configure the following before Setting this bit, and must not modify them while this bit is Set; otherwise, the result is undefined:</p> <ul style="list-style-type: none"> • Selected Algorithm (below) • PCRC Enable • Requester ID Limit in IDE RID Association Register 1 • Requester ID Base , and Segment Base if applicable, in IDE RID Association Register 2 • V bit in IDE RID Association Register 2 <p>If this bit is Set when the V bit is Clear, the IDE Stream must transition to Insecure. When Cleared, must immediately transition the Stream to Insecure.</p> <p>It is strongly recommended that the IDE Address Association Registers, and the Default Stream bit (if applicable), also be programmed prior to Setting this bit.</p> <p>Default value is 0b.</p> | RW |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|-------------------------|--|--------------------------|
| ↑↓1↑ ECN: Base 6.3 XT△↔ | <p>↑↓XT Enable – Controls the generation and processing of the XT bit in IDE TLPs associated with this Stream (see § Section 6.33.4).↑</p> <p>↑↓Must be configured while Selective IDE Stream Enable is Clear, during which time this bit is RW . When Selective IDE Stream Enable is set to 1b, the setting is sampled, and this field becomes RO with reads returning the sampled value.↑</p> <p>↑↓RsvdP if XT Supported bit is Clear.↑</p> <p>↑↓Default value is 0b.↑</p> | ↑↓RW / RO / RsvdP↑ |
| 3:2 | <p>Tx Aggregation Mode NRP – If Aggregation Supported is Set then this field selects the level of aggregation for Transmitted Non-Posted Requests for this Stream, encoded as:</p> <ul style="list-style-type: none"> 00b No aggregation 01b Up to 2 Non-Posted Requests 10b Up to 4 Non-Posted Requests 11b Up to 8 Non-Posted Requests <p>Reserved If Aggregation Supported is Clear.</p> <p>Default value is 00b</p> | RW / RsvdP |
| 5:4 | <p>Tx Aggregation Mode PR – If Aggregation Supported is Set then this field selects the level of aggregation for Transmitted Posted Requests for this Stream, encoded as:</p> <ul style="list-style-type: none"> 00b No aggregation 01b Up to 2 Posted Requests 10b Up to 4 Posted Requests 11b Up to 8 Posted Requests <p>Reserved If Aggregation Supported is Clear.</p> <p>Default value is 00b</p> | RW / RsvdP |
| 7:6 | <p>Tx Aggregation Mode CPL – If Aggregation Supported is Set then this field selects the level of aggregation for ↑↓Trasmitted↓ ↑↓Transmitted↑ Completions for this Stream, encoded as:</p> <ul style="list-style-type: none"> 00b No aggregation 01b Up to 2 Completions 10b Up to 4 Completions 11b Up to 8 Completions <p>Reserved If Aggregation Supported is Clear.</p> <p>Default value is 00b</p> | RW / RsvdP |
| 8 | <p>PCRC Enable – When Set, Transmitted IDE TLPs associated with this Stream that include P content must include PCRC, and Received TLPs must be checked for PCRC failure.</p> <p>Reserved if PCRC Supported is Clear.</p> <p>Default value is 0b.</p> | RW / RsvdP |
| 9 | <p>Selective IDE for Configuration Requests Enable –</p> <p>For Root Ports, if Selective IDE for Configuration Requests Supported is Set, then this bit, when Set, must cause the Port to transmit as IDE TLPs associated with this Selective IDE Stream all Configuration Requests for which the destination RID is greater than or</p> | RW RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|-----------------|
| | <p>equal to the RID Base and less than or equal to the RID Limit in the Selective IDE RID Association Register Block .</p> <p>For Ports other than Root Ports, this bit is Reserved.</p> <p>If Selective IDE for Configuration Requests Supported is Clear, this bit is Reserved.</p> <p>Default value is 0b.</p> | |
| 13:10 | <p>Partial Header Encryption Mode – Selects the mode to be used for partial header encryption of IDE TLPs for this IDE Stream. Must be programmed to the same value in both the Partner Ports. Must be configured while Selective IDE Stream Enable is Clear. When Selective IDE Stream Enable is Set, the setting is sampled, and this field becomes RO with reads returning the sampled value.</p> <ul style="list-style-type: none"> 0000b No partial header encryption 0001b Address[17:2] Encrypted, and, if present, the First DW BE and Last DW BE fields 0010b Address[25:2] Encrypted, and, if present, the First DW BE and Last DW BE fields 0011b Address[33:2] Encrypted, and, if present, the First DW BE and Last DW BE fields 0100b Address[41:2] Encrypted, and, if present, the First DW BE and Last DW BE fields Others Reserved <p>If Partial Header Encryption Supported is Clear, this field is Reserved.</p> | RW / RO / RsvdP |
| 18:14 | <p>Selected Algorithm – Selects the algorithm to be used for securing IDE TLPs for this IDE Stream. Must be programmed to the same value in both Partner Ports. Must be configured while Selective IDE Stream Enable is Clear. When Selective IDE Stream Enable is Set, the setting is sampled, and this field becomes RO with reads returning the sampled value.</p> <ul style="list-style-type: none"> 0 0000b AES-GCM 256 key size, 96b MAC Others Reserved | RW / RO |
| 21:19 | <p>TC – System firmware/software must program this field to indicate the TC associated with this Selective IDE Register block.</p> <p>Default value is 000b</p> | RW |
| 22 | <p>Default Stream – When Set, TLPs using the Traffic Class indicated in the TC field are associated with this Stream, unless the TLP matches some other Selective IDE Stream for the indicated TC. A Default Stream must have the hierarchy domain's Root Port as its Partner Port; otherwise, the result is undefined</p> <p>It is not permitted to configure more than one Default Stream to be associated with the same TC. If this is done, hardware must select one of the Streams to be associated with the TC – the selection is implementation specific.</p> <p>Applicable for Endpoint Upstream Ports only. Reserved for other Port types.</p> <p>Default value is 0b.</p> | RW / RsvdP |
| 23 | <p>TEE-Limited Stream – When Set, requires that, for Requests, only those that ↓↓have↓ ↑↑meet↑ the ↓↓T bit Set↓ ↑↑criteria defined in § Section 6.33.4↑ are permitted to be associated with this Stream.</p> <p>ECN: Base 6.3 XT△↔</p> <p>Must be configured while Selective IDE Stream Enable is Clear, during which time this bit is RW . When Selective IDE Stream Enable is Set ↓↓set↓ to 1b, the setting is sampled, and</p> | RW / RO / RsvdP |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | <p>this field bit becomes RO with reads returning the sampled value, during the time when Selective IDE Stream Enable remains Set.</p> <p>Reserved if TEE-Limited Stream Supported is Clear.</p> <p>Default value is 0b.</p> | |
| 31:24 | <p>N↓Stream_ID↓ ↑↑Stream_ID↑ – Indicates the N↓Stream_ID↓ ↑↑Stream_ID↑ associated with this Selective IDE Stream. Software must program the same N↓Stream_ID↓ ↑↑Stream_ID↑ into both Ports associated with a given Selective IDE Stream. Default value is 00h.</p> | RW |

7.9.26.5.3 Selective IDE Stream Status Register §

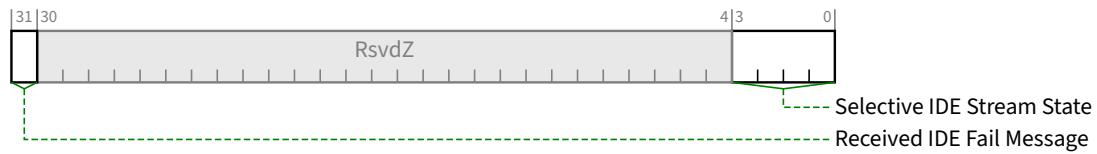


Figure 7-385 Selective IDE Stream Status Register §

Table 7-337 Selective IDE Stream Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 3:0 | <p>Selective IDE Stream State – When Selective IDE Stream Enable is Set, this field indicates the state of the Port. Encodings:</p> <ul style="list-style-type: none"> 0000b Insecure 0010b Secure Others Reserved – Software must handle reserved values as indicating unknown state <p>When Selective IDE Stream Enable is Clear, the value of this field must be 0000b.</p> | RO |
| 31 | Received IDE Fail Message – When Set, indicates that one or more IDE Fail Message(s) have been Received for this Stream. | RW1C |

7.9.26.5.4 Selective IDE RID Association Register Block §

A Selective IDE RID Association register must consist of one IDE RID Association Register 1 followed by one IDE RID Association Register 2.

7.9.26.5.4.1 IDE RID Association Register 1 §

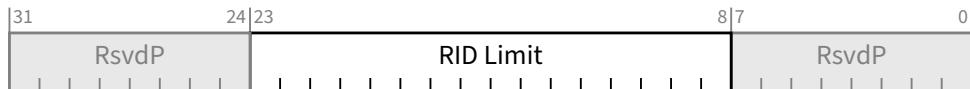


Figure 7-386 IDE RID Association Register 1 (Offset +00h) §

Table 7-338 IDE RID Association Register 1 (Offset +00h) §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 23:8 | <p>RID Limit – Indicates the highest value RID in the range associated with this Stream ID at the IDE Partner Port.</p> <p>The Segment Number associated with this field is contained in <u>Segment Base</u> in the <u>IDE RID Association Register 2</u>.</p> | RW |

7.9.26.5.4.2 IDE RID Association Register 2 §

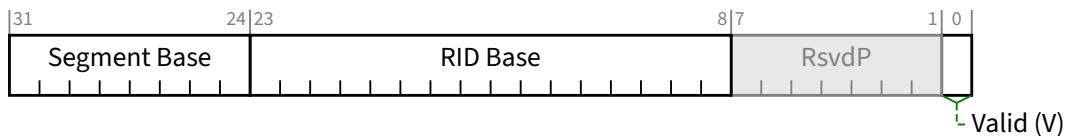


Figure 7-387 IDE RID Association Register 2 (Offset +04h) §

Table 7-339 IDE RID Association Register 2 (Offset +04h) §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Valid (V) – When Set, indicates the <u>Segment Base</u>, <u>RID Base</u> and <u>RID Limit</u> fields have been programmed.</p> <p>Default is 0b</p> | RW |
| 23:8 | <p>RID Base – Indicates the lowest value RID in the range associated with this Stream ID at the IDE Partner Port.</p> <p>The Segment Number associated with this field is contained in <u>Segment Base</u>.</p> | RW |
| 31:24 | <p>Segment Base – In Flit Mode, Indicates the Segment value associated with this Stream ID at the IDE Partner Port.</p> <p>Reserved if Flit Mode is not supported.</p> <p>If this Selective IDE Stream is within an FM subtree whose <u>Segment Captured</u> bits are Clear, software must set this field to 00h, regardless of the Segment Number value associated with the subtree's RP.</p> <p>Default value is 00h.</p> | RW / RsvdP |

7.9.26.5.5 Selective IDE Address Association Register Block §

A Selective IDE Address Association register must consist of one IDE Address Association Register 1 , followed by one IDE Address Association Register 2 , followed by one IDE Address Association Register 3 .

7.9.26.5.5.1 IDE Address Association Register 1 §

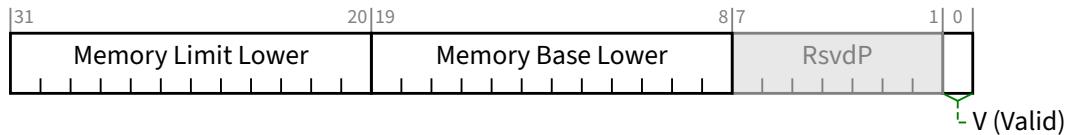


Figure 7-388 IDE Address Association Register 1 (Offset +00h) §

Table 7-340 IDE Address Association Register 1 (Offset +00h) §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:20 | Memory Limit Lower – Corresponds to Address bits [31:20]. Address bits [19:0] are implicitly F_FFFFh. | RW |
| 19:8 | Memory Base Lower – Corresponds to Address bits [31:20]. Address[19:0] bits are implicitly 0_0000h. | RW |
| 0 | V (Valid) – When Set, indicates this IDE Stream Association Block is valid, that the address range defined by Memory Base and Memory Limit corresponding to a range of memory addresses assigned to the IDE Partner Port, and that all Transmitted Address Routed TLPs within this address range must be associated with this IDE Stream, subject to rules stated in § Section 6.33.4. Hardware behavior is undefined if overlapping address ranges are assigned for different IDE Streams. Default is 0b | RW |

7.9.26.5.5.2 IDE Address Association Register 2 §

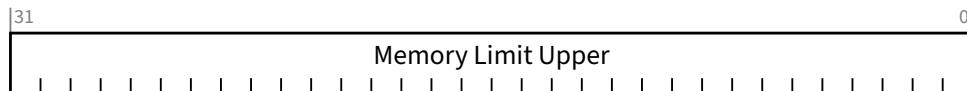


Figure 7-389 IDE Address Association Register 2 (Offset +04h) §

Table 7-341 IDE Address Association Register 2 (Offset +04h) §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 31:0 | Memory Limit Upper – Corresponds to Address bits [63:32] | RW |

7.9.26.5.5.3 IDE Address Association Register 3 §

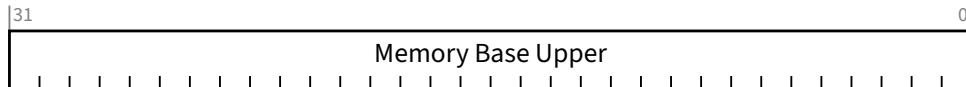


Figure 7-390 IDE Address Association Register 3 (Offset +04h) §

Table 7-342 IDE Address Association Register 3 (Offset +04h) §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 31:0 | Memory Base Upper – Corresponds to Address bits [63:32] | RW |

7.9.27 Null Capability §

The Null Capability is a capability structure in PCI-compatible Configuration Space (first 256 bytes) as shown in § Figure 7-391 .

The Null Capability contains no registers. This capability is present in the linked list (Next Capability Pointer), but should otherwise be ignored by software. The layout of the information is shown in § Figure 7-391 .

A single PCI Express Function is permitted to contain multiple Null Capability structures.

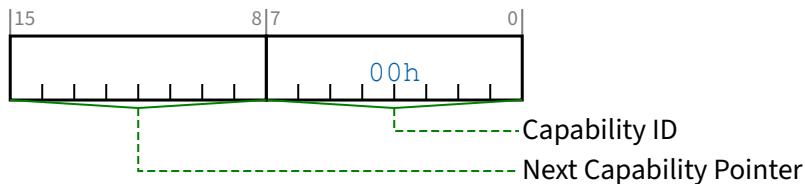


Figure 7-391 Null Capability §

Table 7-343 Null Capability §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 7:0 | Capability ID - Indicates the PCI Express Capability structure. This field must return a Capability ID of 00h indicating that this is a Null Capability structure. | RO |
| 15:8 | Next Capability Pointer - This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities. | RO |

7.9.28 Null Extended Capability §

The Null Extended Capability is an optional Extended Capability that is permitted to be implemented by any PCI Express Function or RCRB . This capability contains no registers. This capability is present in the linked list (Next Capability Offset) but should otherwise be ignored by software.

A single PCI Express Function or RCRB is permitted to contain multiple Null Extended Capability structures.

§ Figure 7-392 details allocation of register fields in the Null Extended Capability ; § Table 7-344 provides the respective bit definitions. The Extended Capability ID for the Null Extended Capability is 0000h .

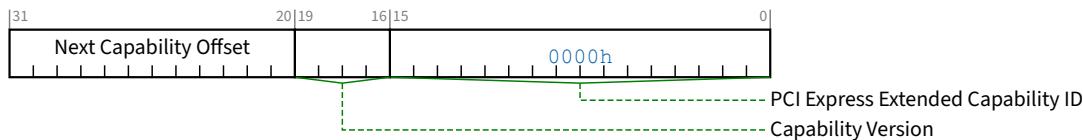


Figure 7-392 Null Extended Capability §

Table 7-344 Null Extended Capability §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Null Extended Capability is 0000h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. This field is permitted to contain any value. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.9.29 Streamlined Virtual Channel Extended Capability (SVC) §

The Streamlined Virtual Channel (SVC) Extended Capability is an optional Extended Capability required for Ports that support capabilities associated with this structure, including UIO. It is permitted, but not required, for Functions in a Port to implement the SVC Extended Capability as well as the MFVC Extended Capability and/or VC Capabilities. See § Section 6.3.5 .

UIO requires the use of the SVC capability and is not supported by the VC or MFVC capabilities. UIO is supported only in Flit Mode, but in Non-Flit Mode the SVC capability can be used by non-UIO traffic.

For an Upstream Port, the SVC Extended Capability structure is permitted to be implemented only in Function 0, and that instance applies to all Functions associated with that Port. The SVC Extended Capability structure is permitted to be implemented in any Downstream Port, or in an RCRB. If the SVC Extended Capability structure is implemented in a USP

containing one or more Switch USP Functions, it must be implemented in all associated Switch DSP Functions. A Root Complex is permitted to implement the Extended Capability structure in some Root Ports and not others.

The number of (extended) Virtual Channels is indicated by the SVC Extended VC Count field in the SVC Port VC Capability Register 1. Software must interpret this field to determine the availability of extended SVC Resource registers.

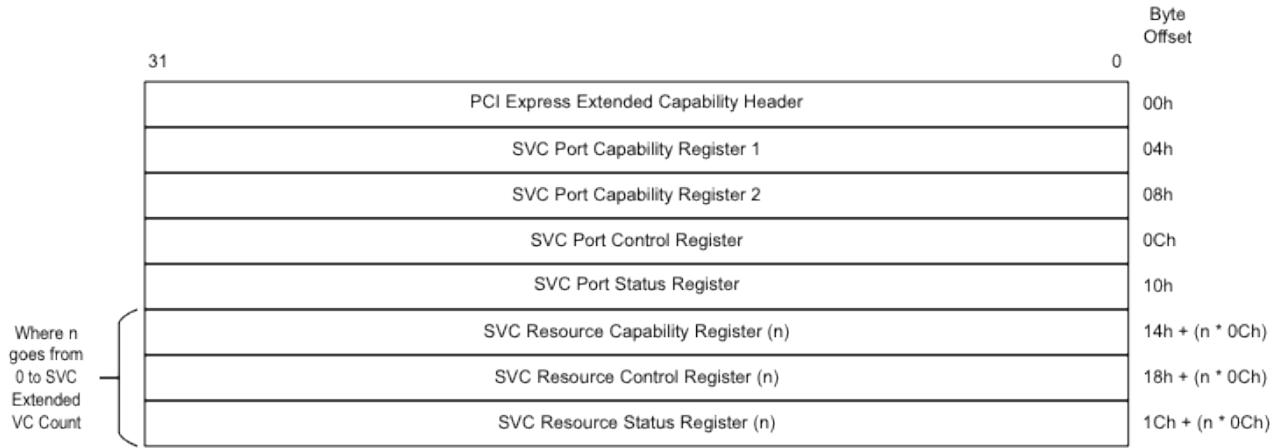


Figure 7-393 Streamlined Virtual Channel Extended Capability Structure §

7.9.29.1 Streamlined Virtual Channel Extended Capability Header (Offset 00h) §

§ Figure 7-394 details allocation of register fields in the Streamlined Virtual Channel Extended Capability Header; § Table 7-345 provides the respective bit definitions.

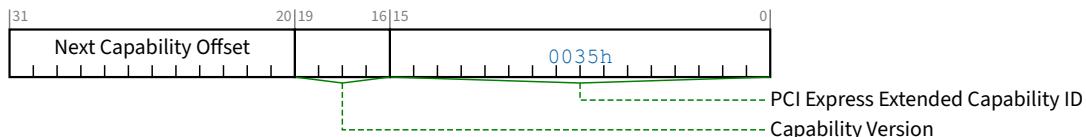


Figure 7-394 Streamlined Virtual Channel Extended Capability Header §

Table 7-345 Streamlined Virtual Channel Extended Capability Header §

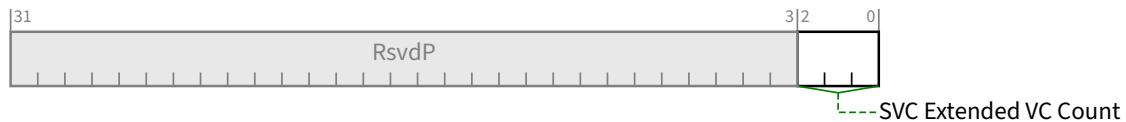
| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Streamlined Virtual Channel Extended Capability is 0035h . | 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. | RO |

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. | |

7.9.29.2 SVC Port Capability Register 1 (Offset 04h) [§](#)

The SVC Port Capability Register 1 describes the configuration of the Virtual Channels associated with a PCI Express Port.

[§ Figure 7-395](#) details allocation of register fields in the SVC Port Capability Register 1; [§ Table 7-346](#) provides the respective bit definitions.



[Figure 7-395 SVC Port Capability Register 1](#) [§](#)

[Table 7-346 SVC Port Capability Register 1](#) [§](#)

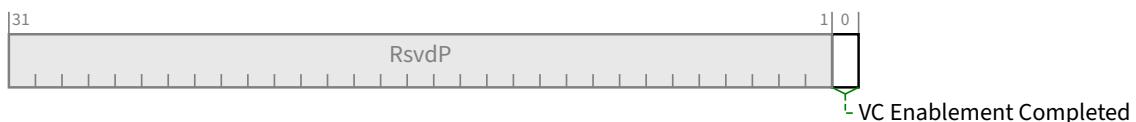
| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | SVC Extended VC Count – Indicates the number of Virtual Channels supported in addition to VC0. This value indicates the number of SVC Resource Capability, Control, and Status registers present in this structure in addition to those for VC0. The minimum value of this field is 0, for devices that only support the default VC. | HwInit |

7.9.29.3 SVC Port Capability Register 2 (Offset 08h) [§](#)

This register is RsvdP .

7.9.29.4 SVC Port Control Register (Offset 0Ch) [§](#)

[§ Table 7-347](#) details allocation of register fields in the SVC Port Control Register; [§ Table 7-347](#) provides the respective bit definitions.



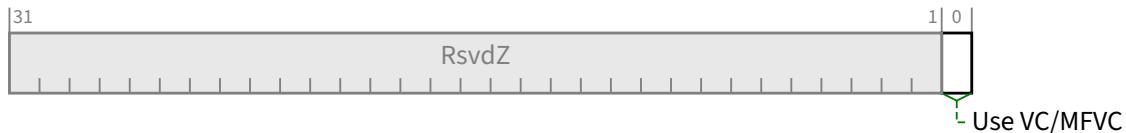
[Figure 7-396 SVC Port Control Register](#) [§](#)

Table 7-347 SVC Port Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>VC Enablement Completed – Setting this bit indicates that software has completed enabling all VCs that are to be used by the Port.</p> <p>Setting this bit is optional. If this bit remains Clear, the Port and all enabled VCs must operate correctly.</p> <p>Default value of this bit is 0b.</p> | RW |

7.9.29.5 SVC Port Status Register (Offset 10h) §

§ Table 7-348 details allocation of register fields in the [↑↓SVC Port Status Register;](#) [↑↑SVC Port Status Register ;↑](#)
 § Table 7-348 provides the respective bit definitions.

*Figure 7-397 SVC Port Status Register §**Table 7-348 SVC Port Status Register §*

| Bit Location | Register Description | Attributes |
|--------------|--|---------------|
| 0 | <p>Use VC/MFVC – This bit enables or disables multiple specific VCs in SVC, VC, and MFVC capabilities as required for § Section 6.3.5.</p> <p>When this bit is Set, the VC Enable bit for VC0 in each VC/MFVC capability (VC Resource Control Register or MFVC VC Resource Control Register) must be Set, and the SVC VC Enable bit for each VC in its SVC Resource Control Register must be Clear.</p> <p>When this bit is Cleared, the VC Enable bit for each VC resource in each VC/MFVC capability (VC Resource Control Register or MFVC VC Resource Control Register) must immediately be Cleared, and the SVC VC Enable bit for VC0 in its SVC Resource Control Register must immediately be Set.</p> <p>If this Port implements any VC or MFVC capabilities, this bit must have a default value of 1b, and if Cleared, it must remain Clear until the next Conventional Reset.</p> <p>If this Port implements no VC or MFVC capabilities, this bit must be 0b.</p> | RW1C / HwInit |

7.9.29.6 SVC Resource Capability Register §

§ Figure 7-398 details allocation of register fields in the SVC Resource Capability Register; § Table 7-349 provides the respective bit definitions.

Base 6.4 vs Base 6.3

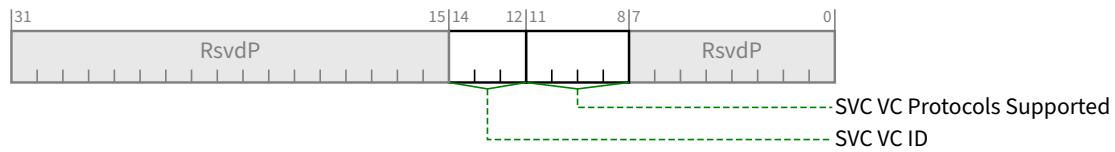


Figure 7-398 SVC Resource Capability Register §

Table 7-349 SVC Resource Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------|
| 11:8 | <p>SVC VC Protocols Supported – Indicates UIO and non-UIO Transaction support for this VC, encoded as:</p> <ul style="list-style-type: none"> 0000b Supports same TLP Types and protocols as VC0 0001b Supports same TLP Types and protocols as VC0, except for those that are restricted by this document to using only VC0 0010b It is permitted to enable UIO on this VC resource; it is not permitted to use non-UIO TLP Types on this VC resource. 0011b It is permitted to enable UIO on this VC resource, or to use as a 0001b VC resource, but not both at the same time. 0100b to 1110b Reserved 1111b Vendor-defined use (outside the scope of this specification) <p>For VC0, must be 0000b. For VC3, if UIO is supported, must be 0010b or 0011b. For VC4, if UIO and VC4 are supported, must be 0010b or 0011b.</p> | HwInit / RsvdP |
| 14:12 | <p>SVC VC ID – This field indicates the VC ID to the VC resource. For the first of SVC Resource Capability Register, this field must be 000b. For other SVC Resource Capability Registers, this field may contain any non-zero value. This field value must be unique across all SVC Resource Capability Registers.</p> | HwInit / RsvdP |

7.9.29.7 SVC Resource Control Register §

§ Figure 7-399 details allocation of register fields in the SVC Resource Control Register; § Table 7-350 provides the respective bit definitions.

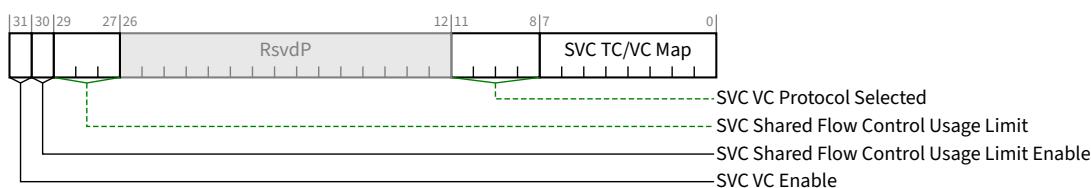


Figure 7-399 SVC Resource Control Register §

Table 7-350 SVC Resource Control Register §

| Bit Location | Register Description | Attributes | | | | | | | | | | | | | | | | |
|-----------------------|---|--------------|--|--------------|---|--------------|-----------------------------|-----------------------|----------|--------------|--|-----------------|-------|-------------|-----|-------------|-------|-----------------|
| 7:0 | <p>SVC TC/VC Map – This field indicates the TCs that are mapped to the VC resource. This field is valid for all Functions.</p> <p>Bit locations within this field correspond to TC values.</p> <p>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 Requests with the TC labels are targeted at the given Link.</p> <p>Bit 0 of this field must be read-only. It must be Set for the default VC0 and Clear for all other enabled VCs.</p> <p>Default value of this field must be consistent with Table 2-46.</p> | RW / RO | | | | | | | | | | | | | | | | |
| 11:8 | <p>SVC VC Protocol Selected – Determines the TLP Types and Protocols to be used with this VC, encoded as:</p> <table> <tr> <td>0000b</td><td>TLP Types and protocols as VC0 (required for VC0 and permitted only for VC0)</td></tr> <tr> <td>0001b</td><td>TLP Types and protocols as VC0, except for those that are restricted by this document to using only VC0</td></tr> <tr> <td>0010b</td><td>UIO TLP Types and protocols</td></tr> <tr> <td>0011b to 1110b</td><td>Reserved</td></tr> <tr> <td>1111b</td><td>Vendor-defined use (outside the scope of this specification)</td></tr> </table> <p>For VC0, must be hardwired to 0000b. Default value is implementation specific.</p> | 0000b | TLP Types and protocols as VC0 (required for VC0 and permitted only for VC0) | 0001b | TLP Types and protocols as VC0, except for those that are restricted by this document to using only VC0 | 0010b | UIO TLP Types and protocols | 0011b to 1110b | Reserved | 1111b | Vendor-defined use (outside the scope of this specification) | RW / RO / RsvdP | | | | | | |
| 0000b | TLP Types and protocols as VC0 (required for VC0 and permitted only for VC0) | | | | | | | | | | | | | | | | | |
| 0001b | TLP Types and protocols as VC0, except for those that are restricted by this document to using only VC0 | | | | | | | | | | | | | | | | | |
| 0010b | UIO TLP Types and protocols | | | | | | | | | | | | | | | | | |
| 0011b to 1110b | Reserved | | | | | | | | | | | | | | | | | |
| 1111b | Vendor-defined use (outside the scope of this specification) | | | | | | | | | | | | | | | | | |
| 29:27 | <p>SVC Shared Flow Control Usage Limit – This field controls what percentage of the available Shared Flow Control a given FC/VC is permitted to consume.</p> <p>This limit is applied independently for each Flow Control credit type. For example, if this field contains 101b and SVC Shared Flow Control Usage Limit Enable is Set, a Posted TLP may not pass the Tx Gate if doing so would cause that VC to consume more than 62.5% of the available Shared Posted Header credits or if doing so would cause that VC to consume more than 62.5% of the available Shared Data credits.</p> <p>If SVC Shared Flow Control Usage Limit Enable is Clear, this field must be ignored, and this VC is permitted to consume all of the shared credits, unless the Transmitter has implementation-specific policy mechanisms to constrain shared credit use.</p> <p>When SVC Shared Flow Control Usage Limit Enable is Set, and this field contains 000b, this VC is not permitted to consume any shared credits.</p> <p>Behavior is undefined when all VCs have SVC Shared Flow Control Usage Limit Enable Set and the sum of the SVC Shared Flow Control Limit values for all VCs is less than 100%.</p> <p>Encodings are:</p> <table> <tr> <td>000b</td><td>0%</td></tr> <tr> <td>001b</td><td>12.5%</td></tr> <tr> <td>010b</td><td>25%</td></tr> <tr> <td>011b</td><td>37.5%</td></tr> <tr> <td>100b</td><td>50%</td></tr> <tr> <td>101b</td><td>62.5%</td></tr> <tr> <td>110b</td><td>75%</td></tr> <tr> <td>111b</td><td>87.5%</td></tr> </table> <p>Behavior is undefined if this field changes value while SVC VC Enable and SVC Shared Flow Control Usage Limit Enable are both Set.</p> <p>When SVC Extended VC Count is 0, this field is permitted to be hardwired to any value.</p> <p>When this field is RW, the default value is implementation specific.</p> | 000b | 0% | 001b | 12.5% | 010b | 25% | 011b | 37.5% | 100b | 50% | 101b | 62.5% | 110b | 75% | 111b | 87.5% | RW / RO / RsvdP |
| 000b | 0% | | | | | | | | | | | | | | | | | |
| 001b | 12.5% | | | | | | | | | | | | | | | | | |
| 010b | 25% | | | | | | | | | | | | | | | | | |
| 011b | 37.5% | | | | | | | | | | | | | | | | | |
| 100b | 50% | | | | | | | | | | | | | | | | | |
| 101b | 62.5% | | | | | | | | | | | | | | | | | |
| 110b | 75% | | | | | | | | | | | | | | | | | |
| 111b | 87.5% | | | | | | | | | | | | | | | | | |

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------------------|
| 30 | <p>SVC Shared Flow Control Usage Limit Enable – When Set, this bit enables use of control of Shared Flow Control consumption at the transmitter for this Virtual Channel.</p> <p>Behavior is undefined if the value of this bit changes while <u>SVC VC Enable</u> is Set.</p> <p>This bit is <u>RsvdP</u> when Flit Mode Supported is Clear.</p> <p>When <u>SVC Extended VC Count</u> is 0, this bit is permitted to be hardwired to 0b.</p> <p>When this bit is <u>RW</u>, the default value is implementation specific.</p> | <u>RW / RO</u> / <u>RsvdP</u> |
| 31 | <p>SVC VC Enable – This bit, when Set, enables this Virtual Channel. The Virtual Channel is disabled when this bit is cleared.</p> <p>Software must use the SVC VC Negotiation Pending bit to check whether the VC negotiation is complete.</p> <p>For VC0, the attribute is <u>RO</u>, and this bit must always have the opposite value from the <u>Use VC/MFVC</u> bit in the <u>N↓SVC Port Status Register</u>. See § <u>Section 6.3.5</u>.</p> <p>For other VCs, the default value of this bit is 0b and the attribute is <u>RW</u>.</p> <p>To enable a Virtual Channel in a Port using SVC mechanisms, the SVC VC Enable bit for that Virtual Channel must be Set. The corresponding Virtual Channel in the Link partner Port must be enabled as well, and that Virtual Channel may be in an SVC, VC, or MFVC capability. To disable a Virtual Channel, the Virtual Channel must be disabled in both components on the Link. Software must ensure that no traffic is using a Virtual Channel at the time it is disabled. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.</p> | <u>RW / RO</u> |

7.9.29.8 SVC Resource Status Register §

§ Table 7-351 details allocation of register fields in the VC Resource Status Register; § Table 7-351 provides the respective bit definitions.

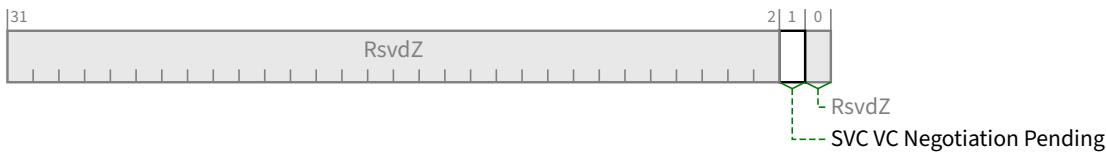


Figure 7-400 SVC Resource Status Register §

Table 7-351 SVC Resource Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 1 | <p>SVC VC Negotiation Pending – This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state. This bit is valid for all Functions.</p> <p>The value of this bit is defined only when the Link is in the <u>DL_Active</u> state and the corresponding <u>SVC VC Enable</u> bit is Set.</p> <p>This bit is Set by hardware to indicate that the VC resource has not completed the process of negotiation. This bit is Cleared by hardware after the VC negotiation is complete (on exit from the <u>FC_INIT2</u> state). For VC0, this bit is permitted to be hardwired to 0b.</p> <p>Before using a Virtual Channel, software must check whether the <u>SVC VC Negotiation Pending</u> bits for that Virtual Channel are Clear in both components on the Link.</p> | <u>RO</u> |

7.9.30 MMIO Register Block Locator Extended Capability (MRBL) §

The MMIO Register Block Locator Extended Capability (MRBL) is an optional Extended Capability for discovering register blocks in Memory Space that can be used to exchange various types of data structures between system software and a Function (See § Section 6.35 § Section 6.35).

It is permitted to implement the MRBL Extended Capability in any type of Function. A single PCI Express Function is permitted to contain at most one instance of this capability.

The number of register blocks included the MRBL structure is described in the MRBL Capabilities Register (§ Section 7.9.30.2 § Section 7.9.30.2). A Function that implements the MRBL Extended Capability shall support at least one MRBL Locator Register (§ Section 7.9.30.3).

Each register block is described by a MRBL Locator Register (§ Section 7.9.30.3) to specify the location and type of the registers within Memory Space. Each register block must be contained within the address range covered by the associated BAR. § Figure 7-401 illustrates the MRBL Extended Capability structure.

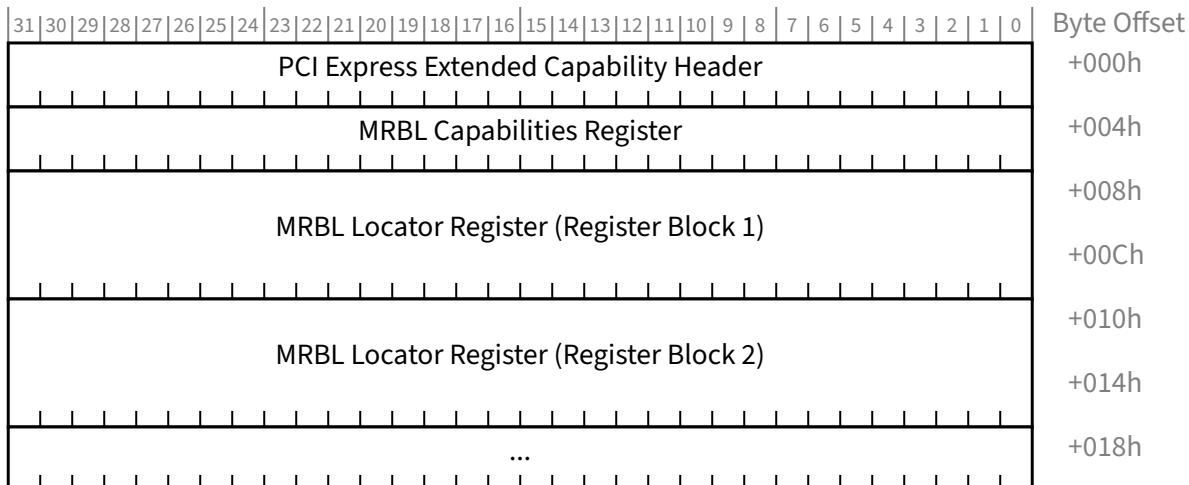


Figure 7-401 MRBL Extended Capability §

7.9.30.1 MRBL Extended Capability Header (Offset 00h) §

§ Figure 7-402 details allocation of register fields in the MRBL Extended Capability Header ; § Table 7-352 provides the respective bit definitions. Refer to § Section 7.6.3 for a description of the PCI Express Extended Capability Header . The Extended Capability ID for the MRBL Extended Capability is 0036h .

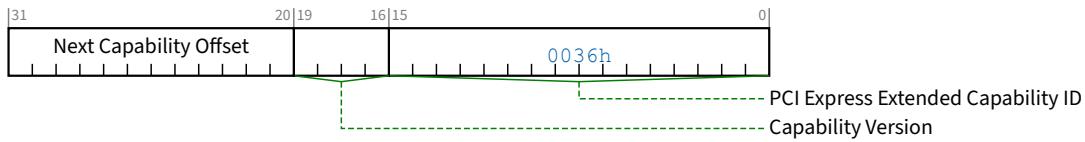


Table 7-352 MRBL Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the MMIO Register Block Locator Extended Capability is 0036h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

7.9.30.2 MRBL Capabilities Register (Offset 04h) §

§ Figure 7-403 details allocation of register fields in the MRBL Capabilities Register ; § Table 7-353 provides the respective bit definitions.



Table 7-353 MRBL Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 11:0 | MRBL Structure Length - This field indicates the overall size of the MRBL Extended Capability in bytes: 08h + (n*08h) where n is the number of MRBL Locator Registers implemented. | HWInit |

7.9.30.3 MRBL Locator Register (Offset Varies) §

§ Figure 7-404 details allocation of register fields in the MRBL Locator Register ; § Table 7-354 provides the respective bit definitions.

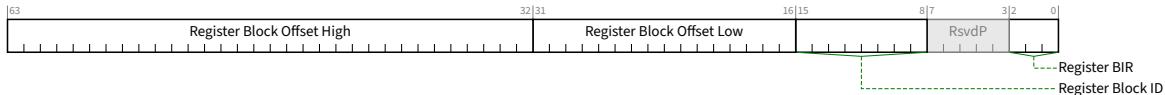


Figure 7-404 MRBL Locator Register §

Table 7-354 MRBL Locator Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 2:0 | <p>Register BIR – Indicates which one of a Function’s Base Address Registers, or entry in the Enhanced Allocation capability with a matching BAR Equivalent Indicator (BEI), is used for this register block.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 0 Base Address Register 10h 1 Base Address Register 14h 2 Base Address Register 18h 3 Base Address Register 1Ch 4 Base Address Register 20h 5 Base Address Register 24h 6 Reserved 7 Reserved <p>For a 64-bit Base Address Register, the Register BIR indicates the lower DWORD. For Functions with Type 1 Configuration Space headers , BIR values 2 through 5 are Reserved.</p> | HWInit |
| 15:8 | <p>Register Block ID – Identifies the type of registers contained in this register block.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> 00h Empty/invalid register block 01h MMIO Capabilities Register Block (MCAP) (§ Section 6.35.1) 02h-FEh Reserved FFh MMIO Designated Vendor-Specific Register Block (MDVS) (§ Section 6.35.2) | HWInit |
| 31:16 | <p>Register Block Offset Low – Contains bits [31:16] of the register block address offset within the BAR/BEI indicated by Register BIR . Offset bits [15:0] are 0000h.</p> <p>The value in this field must be ignored if Register Block ID is 00h.</p> | HWInit |
| 63:32 | <p>Register Block Offset High – Contains bits [63:32] of the register block address offset within the BAR/BEI indicated by Register BIR .</p> <p>The value in this field must be ignored if Register Block ID is 00h.</p> | HWInit |

8. Electrical Sub-Block §

8.1 Electrical Specification Introduction §

Key attributes of the Electrical Specification include:

- Support for NRZ signaling at 2.5, 5.0, 8.0, 16.0, 32.0 GT/s, and PAM4 signaling at 64.0 GT/s data rates
- Support for common and separate independent reference clock architectures
- Support for Spread Spectrum clocking
- Reduced swing mode for lower power Link operation
- In-band Receiver detection and electrical idle detection
- Channel compliance methodology
- Adaptive transmitter equalization and reference Receiver equalization allowing closed eye channel support at 8.0, 16.0, and 32.0 GT/s, and 64.0 GT/s
- Lane margining
- AC coupled channel

Please note that throughout this specification, the term GT/s is used to refer to the number of encoded bits transferred in a second on a direction of a lane. In PAM4 signaling, two bits are encoded in one UI with four voltage levels § Section 4.2.3.1.1. Consequently, the Nyquist frequency is 16 GHz for both 32.0 GT/s NRZ and 64.0 GT/s PAM4.

Because of four voltage levels and reduced amplitude for each voltage level, 64.0 GT/s PAM4 signaling is sensitive to noise and burst errors. The Bit Error Rate (BER), also referred as First Bit Error Rate (FBER) in § Chapter 4. for 64.0 GT/s is 10^{-6} . FBER refers to Bit Error Rate without accounting for any burst error. For 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s data rates, BER of 10^{-12} implicitly assumes FBER of 10^{-12} and do not account for any types of burst error.

The 6.0 version of the PCI Express Electrical Specification is organized into separate sections for the Transmitter, Receiver, the Channel, and the Refclk. In this version most parameters have been regularized such that a common set of parameters is used to define compliance at all data rates.

8.2 Interoperability Criteria §

8.2.1 Data Rates §

A device must support 2.5 GT/s and is not permitted to skip support for any data rates between 2.5 GT/s and the highest supported rate.

8.2.2 Refclk Architectures §

PCIe supports two Refclk data architectures: Common Refclk, and Independent Refclk. These are described in detail in § Section 8.6. A PCIe device may support one or more of these architectures.

8.3 Transmitter Specification §

8.3.1 Measurement Setup for Characterizing Transmitters §

The PCI Express electrical specification references all measurements to the device's pin. However, the pin of a device under test (DUT) is not generally accessible, and the closest accessible point is usually a pair of microwave-type coaxial connectors separated from the DUT pins by several inches of PCB trace, called the breakout channel on a silicon test board. On a test board with many Lanes the minimum breakout channel length is constrained by the need to route to many coaxial connectors. Typically, this limitation holds true for both the Tx and the Rx pins. § Figure 8-1 illustrates a typical test connection to a DUT, showing a single Tx Lane breakout.

A low jitter Refclk source is used when the silicon supports using an external reference clock in order that the jitter measurements for the DUT include only contributions from the Transmitter.

When testing a Transmitter it is desirable to have as many other PCI Express Lanes sending or receiving the compliance pattern as is feasible. Similarly, if the device supports other I/O it should also be sending or receiving on these interfaces. The goal is to have the Tx test environment replicate that found in a real system as closely as possible.

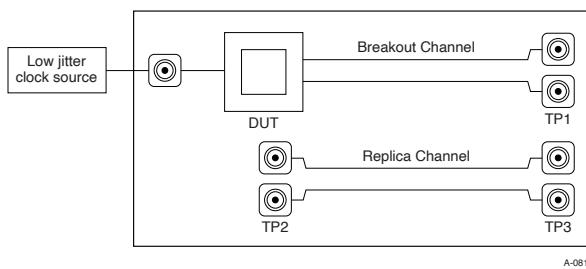


Figure 8-1 Tx Test Board for Non-Embedded Refclk §

The 6.0 version of the Tx specification also includes explicit support for Transmitters utilizing embedded Refclks. In this case the Tx under test is not driven with a low jitter clock source, and both the Tx data and Tx Refclk out must be sampled simultaneously by means of a 2-port measurement as shown in § Figure 8-2 . For more details consult § Section 8.3.5.3 . When an implementation is tested that is configured for the independent reference clock architecture only the data is sampled for both the Non-Embedded and Embedded reference clock cases.

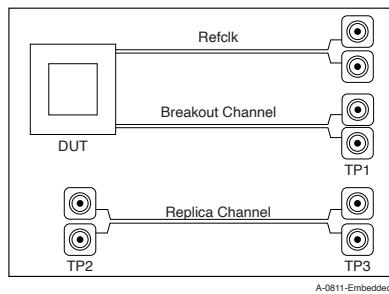


Figure 8-2 Tx Test board for Embedded Refclk §

8.3.1.1 Breakout and Replica Channels §

In order to specify a Transmitter with a uniform set of Tx parameters it is necessary to establish a one-to-one correspondence between what is measurable at TP1 and the corresponding Tx voltage or jitter at the pin. This may be achieved by means of a breakout channel and a replica channel. The replica channel reproduces the electrical characteristics of the breakout channel as closely as possible, matching its length, layer transitions, etc., making it possible to de-embed Tx measurements to the pin of the DUT. All voltage parameters are de-embedded to the pins unless otherwise specified. While the specification does not define precise electrical characteristics for the replica and breakout channels, it is advisable to adhere to the following guidelines:

- Breakout channels should be the same length for each Lane and routed on as few layers as possible, thereby reducing the number of replica channels that need to be built and measured.
- Each routing layer on a test board should have a separate breakout channel where the via and pad structures of the breakout and replica channels on respective layers match as closely as possible.
- Breakout and replica channels should be designed to have an insertion loss of less than 2 dB at the Nyquist frequency for the signaling rate (4 dB at Nyquist if the maximum signaling rate is 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s) and a return loss of greater than 15 dB to Nyquist when measured from either TP2 or TP3, which may require use of low loss dielectric, wide signal traces and back-drilling of break-out vias or use of micro-via technology.
- The impedance targets for the breakout channel are 100 Ω differential and 50 Ω single-ended. For best accuracy the actual breakout channel impedance should be within ±5% of these values. For larger deviations a more complex de-embedding technique may be required.

8.3.2 Voltage Level Definitions §

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

$$V_{DIFFp-p} = (2 * \max |V_{D+} - V_{D-}|) \quad (\text{This applies to a symmetric differential swing})$$

Equation 8-1 $V_{DIFFp-p}$ §

$$V_{TX-AC-CM-PP} = \max(V_{D+} + V_{D-}) / 2 - \min(V_{D+} + V_{D-}) / 2$$

Equation 8-2 $V_{TX-AC-CM-PP}$ §

Note: The maximum value is calculated on a per unit interval evaluation with unit interval boundaries determined by the behavioral CDR. The maximum function as described is implicit for all peak-to-peak and peak common mode equations throughout the rest of this chapter.

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

An example waveform is shown in § Figure 8-3 . In this waveform the differential voltage (defined as D+ - D-) is approximately 800 mVPP, and the single-ended voltage for both D+ and D- is approximately 400 mVPP for each. Note that while the center crossing point for both D+ and D- is nominally at 200 mV, the corresponding crossover point for the differential voltage is at 0.0 V.

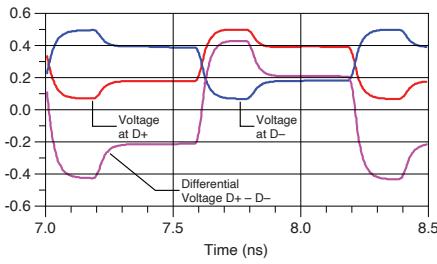


Figure 8-3 Single-ended and Differential Levels §

8.3.3 Tx Voltage Parameters §

Tx voltage parameters include equalization coefficients, equalization presets, and min/max voltage swings.

8.3.3.1 2.5 and 5.0 GT/s Transmitter Equalization §

Tx equalization at 2.5 and 5.0 GT/s is only de-emphasis. Tx equalization de-emphasis values at 2.5 and 5.0 GT/s are measured using the average ratio of transition to non-transition average eye amplitude at the 0.5 UI location using 500 repetitions of the compliance pattern.

8.3.3.2 8.0, 16.0, 32.0, and 64.0 GT/s Transmitter Equalization §

Tx voltage swing and equalization presets at 8.0, 16.0, 32.0, and 64.0 GT/s are measured by means of a low frequency pattern within the compliance pattern. The pattern consists of a sequence of 64 zeros followed by 64 ones, for 8.0, 16.0, and 32.0 GT/s and it consists of a sequence of 64 voltage level 0's followed by 64 voltage level 3's for 64.0 GT/s. The low frequency pattern permits an accurate measurement of voltage since ISI effects will have decayed and the signal will have approached a steady state. 8.0, 16.0, 32.0, and 64.0 GT/s transmitters must implement a coefficient-based equalization mode in order to support fine grained control over Tx equalization resolution. Additionally, a Transmitter must support a specified number of presets that give a coarser control over Tx equalization resolution. Both coefficient space and preset space are controllable via messaging from the Receiver via an equalization procedure. The equalization procedure operates on the same physical path as normal signaling and is implemented via extensions to the existing protocol Link layer.

All 8.0, 16.0, 32.0, and 64.0 GT/s Transmitters must implement support for the equalization procedure, whereas 8.0, 16.0, 32.0, and 64.0 GT/s Receivers may optionally make use of requests for the Transmitter on the Link partner to update Transmitter equalization. Details of the equalization procedure may be found in the Physical Layer Logical Block chapter.

Tx equalization coefficients for 8.0, 16.0, and 32.0 GT/s are based on the following FIR filter relationship as shown in § Figure 8-4 . Equalization coefficients are subject to constraints limiting their max swing to ±unity with c_{-1} and c_{+1} being zero or negative. The inclusion of the unity condition means that only two of the three coefficients need to be specified to fully define $v_{out,n}$. In this specification for 8.0, 16.0, and 32.0 GT/s the two coefficients so specified are c_{-1} and c_{+1} , where c_0 is implied. Note that the coefficient magnitude is not the same as the Tx voltage swing magnitude.

Tx equalization coefficients for 64.0 GT/s are based on the following FIR filter relationship as shown in § Figure 8-5 . Equalization coefficients are subject to constraints limiting their max swing to \pm unity with c_{-2} being zero or positive, c_{-1} and c_{+1} being zero or negative.

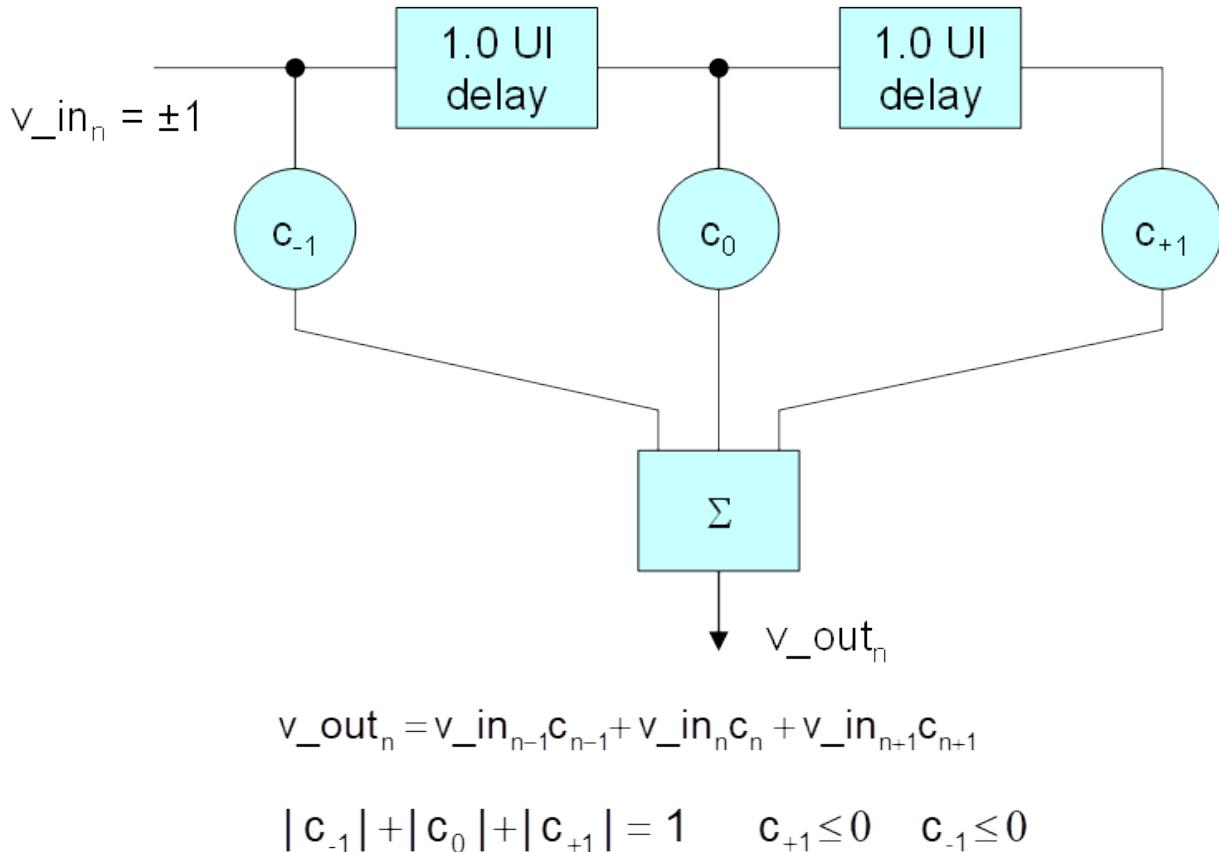


Figure 8-4 Tx Equalization FIR Representation for 8.0, 16.0, and 32.0 GT/s §

Base 6.4 vs Base 6.3

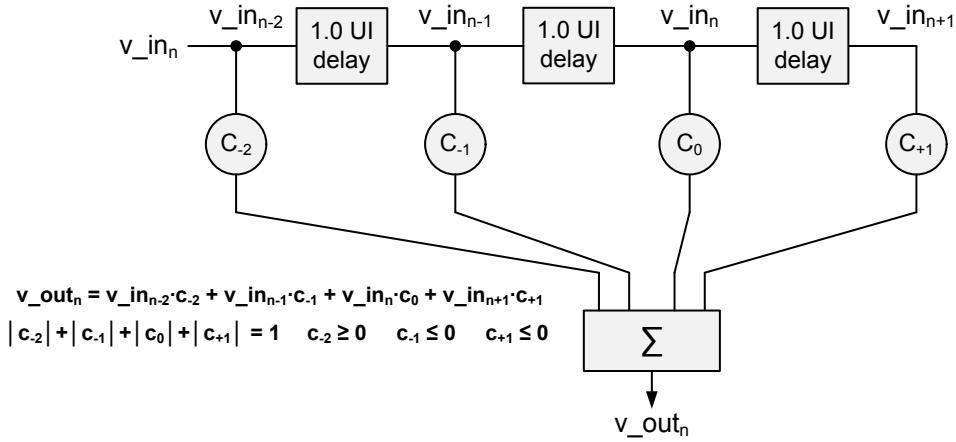


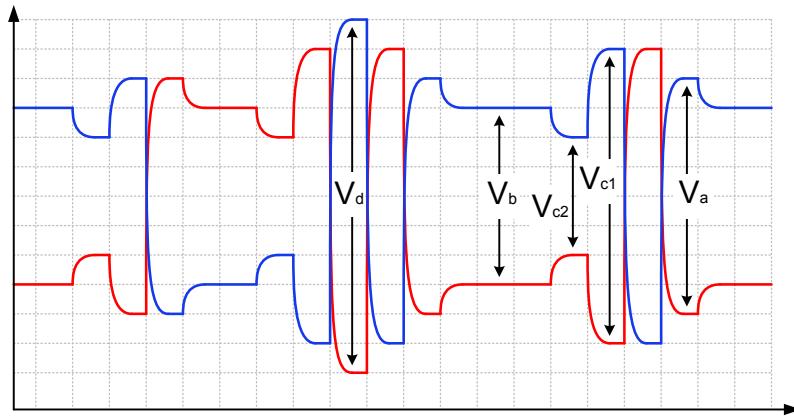
Figure 8-5 Tx Equalization FIR Representation for 64.0 GT/s §

8.3.3.3 Tx Equalization Presets for 8.0, 16.0, 32.0, and 64.0 GT/s §

When operating at 8.0 GT/s, 16.0 GT/s and 32.0 GT/s the Tx must support the full range of presets given in § Table 8-1 . When operating at 64.0 GT/s, the Tx must support the full range of presets given in § Table 8-2 . The data rate dependent encoding of presets has been defined in § Section 4.2.4.2 . Presets are defined in terms of ratios, relating the pre-cursor and post-cursor equalization voltages. The pre-cursors (V_{C1}) and (V_{C2}) are referred to as pre-shoots, while the post-cursor (V_b) is referred to as de-emphasis. This convention permits the specification to retain the existing 2.5 GT/s and 5.0 GT/s definitions for Tx equalization, where only de-emphasis is defined, and it allows pre-shoots and de-emphasis to be defined such that each is independent of the other. The tolerances in § Table 8-1 also apply to 2.5 and 5.0 GT/s de-emphasis. The maximum swing, V_d , is also shown to illustrate that, when c_{+1} , c_{-2} , and c_{-1} are non-zero, the swing of V_a does not reach the maximum as defined by V_d . § Figure 8-6 is shown as an example of transmitter equalization, but it is not intended to represent the signal as it would appear for measurement purposes. The high frequency nature of PCIe signaling makes measurement of single UI pulse heights impractical.

The presets defined in § Table 8-1 and § Table 8-2 are numbered to match the designations in § Table 4-34 .

Base 6.4 vs Base 6.3



| | | | |
|-------------|-----------------------------|--|--|
| De-emphasis | $= 20\log_{10}(V_b/V_a)$ | $V_a = (+c_{-2} + c_{-1} + c_0 - c_{+1})$ | $c_{-2} \geq 0$ |
| Pre-Shoot 1 | $= 20\log_{10}(V_{c1}/V_b)$ | $V_b = (+c_{-2} + c_{-1} + c_0 + c_{+1})$ | $c_{-1} \leq 0$ |
| Pre-Shoot 2 | $= 20\log_{10}(V_{c2}/V_b)$ | $V_{c1} = (+c_{-2} - c_{-1} + c_0 + c_{+1})$ | $c_0 \geq 0$ |
| Boost | $= 20\log_{10}(V_d/V_b)$ | $V_{c2} = (-c_{-2} + c_{-1} + c_0 + c_{+1})$ | $c_{+1} \leq 0$ |
| | | $V_d = (+c_{-2} - c_{-1} + c_0 - c_{+1})$ | $ c_{-2} + c_{-1} + c_0 + c_{+1} = 1$ |

Figure 8-6 Definition of Tx Voltage Levels and Equalization Ratios §

§ Table 8-1 lists the values for presets; at 8.0 GT/s, 16.0 GT/s and 32.0 GT/s. All preset values must be supported for full swing signaling.

Table 8-1 Tx Preset Ratios and Corresponding Coefficient Values for 8.0, 16.0, and 32.0 GT/s §

| Preset # | Preshoot 2 (dB) | Preshoot 1 (dB) | De-emphasis (dB) | c_{-2} | c_{-1} | c_{+1} | V_a/V_d | V_b/V_d | V_{c1}/V_d | V_{c2}/V_d |
|------------|-----------------|-----------------|-------------------|----------|----------|----------|-----------|-----------|--------------|--------------|
| P4 | 0.0 | 0.0 ± 1 dB | 0.0 ± 1 dB | 0.000 | 0.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| P1 | 0.0 | 0.0 ± 1 dB | -3.5 ± 1 dB | 0.000 | 0.000 | -0.167 | 1.000 | 0.666 | 0.666 | 0.666 |
| P0 | 0.0 | 0.0 ± 1 dB | -6.0 ± 1.5 dB | 0.000 | 0.000 | -0.250 | 1.000 | 0.500 | 0.500 | 0.500 |
| P9 | 0.0 | 3.5 ± 1 dB | 0.0 ± 1 dB | 0.000 | -0.167 | 0.000 | 0.666 | 0.666 | 1.000 | 0.666 |
| P8 | 0.0 | 3.5 ± 1 dB | -3.5 ± 1 dB | 0.000 | -0.125 | -0.125 | 0.750 | 0.500 | 0.750 | 0.500 |
| P7 | 0.0 | 3.5 ± 1 dB | -6.0 ± 1.5 dB | 0.000 | -0.100 | -0.200 | 0.800 | 0.400 | 0.600 | 0.400 |
| P5 | 0.0 | 1.9 ± 1 dB | 0.0 ± 1 dB | 0.000 | -0.100 | 0.000 | 0.800 | 0.800 | 1.000 | 0.800 |
| P6 | 0.0 | 2.5 ± 1 dB | 0.0 ± 1 dB | 0.000 | -0.125 | 0.000 | 0.750 | 0.750 | 1.000 | 0.750 |
| P3 | 0.0 | 0.0 ± 1 dB | -2.5 ± 1 dB | 0.000 | 0.000 | -0.125 | 1.000 | 0.750 | 0.750 | 0.750 |
| P2 | 0.0 | 0.0 ± 1 dB | -4.4 ± 1.5 dB | 0.000 | 0.000 | -0.200 | 1.000 | 0.600 | 0.600 | 0.600 |
| P10 | 0.0 | 0.0 ± 1 dB | Note 2 | 0.000 | 0.000 | Note 2 | 1.000 | Note 2 | Note 2 | Note 2 |

Notes:

| Preset # | Preshoot 2 (dB) | Preshoot 1 (dB) | De-emphasis (dB) | c_{-2} | c_{-1} | c_{+1} | Va/Vd | Vb/Vd | Vc1/Vd | Vc2/Vd |
|----------|-----------------|-----------------|------------------|----------|----------|----------|-------|-------|--------|--------|
|----------|-----------------|-----------------|------------------|----------|----------|----------|-------|-------|--------|--------|

- Reduced swing signaling must implement presets P4, P1, P9, P5, P6, and P3. Full swing signaling must implement all the above presets.
- P10 boost limits are not fixed, since its de-emphasis level is a function of the LF level that the Tx advertises during training (see § Section 4.2.4.1). P10 is used for testing the boost limit of Transmitter at full swing. P1 is used for testing the boost limit of Transmitter at reduced swing.

§ Table 8-2 lists the values for presets at 64.0 GT/s. All preset values must be supported for full swing signaling.

Table 8-2 Tx Preset Ratios and Corresponding Coefficient Values for 64.0 GT/s §

| Preset # | Preshoot 2 (dB) | Preshoot 1 (dB) | De-emphasis (dB) | c_{-2} | c_{-1} | c_{+1} | Va/Vd | Vb/Vd | Vc1/Vd | Vc2/Vd |
|------------|-----------------|-----------------|------------------|----------|----------|----------|-------|--------|--------|--------|
| Q0 | 0.0 ±0.5 dB | 0.0 ±0.5 dB | 0.0 ±0.5 dB | 0.000 | 0.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Q1 | 0.0 ±0.5 dB | 1.6 ±0.5 dB | 0.0 ±0.5 dB | 0.000 | -0.083 | 0.000 | 0.834 | 0.834 | 1.000 | 0.834 |
| Q2 | 0.0 ±0.5 dB | 3.5 ±0.5 dB | 0.0 ±0.5 dB | 0.000 | -0.167 | 0.000 | 0.666 | 0.666 | 1.000 | 0.666 |
| Q3 | 0.0 ±0.5 dB | 0.0 ±0.5 dB | -1.6 ±0.5 dB | 0.000 | 0.000 | -0.083 | 1.000 | 0.834 | 0.834 | 0.834 |
| Q4 | 0.0 ±0.5 dB | 0.0 ±0.5 dB | -3.5 ±0.5 dB | 0.000 | 0.000 | -0.167 | 1.000 | 0.666 | 0.666 | 0.666 |
| Q5 | -1.3 ±0.5 dB | 4.7 ±1.0 dB | 0.0 ±0.5 dB | 0.042 | -0.208 | 0.000 | 0.584 | 0.584 | 1.000 | 0.500 |
| Q6 | -1.6 ±0.5 dB | 3.5 ±0.5 dB | -3.5 ±0.5 dB | 0.042 | -0.125 | -0.125 | 0.750 | 0.500 | 0.750 | 0.416 |
| Q7 | -2.9 ±0.5 dB | 4.7 ±1.0 dB | 0.0 ±0.5 dB | 0.083 | -0.208 | 0.000 | 0.584 | 0.584 | 1.000 | 0.418 |
| Q8 | -3.5 ±0.5 dB | 6.0 ±1.0 dB | 0.0 ±0.5 dB | 0.083 | -0.250 | 0.000 | 0.500 | 0.500 | 1.000 | 0.334 |
| Q9 | -4.4 ±1.0 dB | 6.9 ±1.0 dB | -1.6 ±0.5 dB | 0.083 | -0.250 | -0.042 | 0.500 | 0.416 | 0.916 | 0.250 |
| Q10 | 0.0 ±0.5 dB | 0.0 ±0.5 dB | Note 2 | 0.000 | 0.000 | Note 2 | 1.000 | Note 2 | Note 2 | Note 2 |

Notes:

- Reduced swing signaling must implement presets Q0, Q1, Q2, Q3, and Q4. Full swing signaling must implement all the above presets.
- Q10 boost limits are not fixed, since its de-emphasis level is a function of the LF level that the Tx advertises during training. Q10 is used for testing the boost limit of Transmitter at full swing. Q4 is used for testing the boost limit of Transmitter at reduced swing.

8.3.3.4 Measuring Tx Equalization for 2.5 GT/s and 5.0 GT/s §

Tx equalization de-emphasis values at 2.5 and 5.0 GT/s are measured using the average ratio of transition to non-transition eye heights at the 0.5 UI location using 500 repetitions of the compliance pattern.

8.3.3.5 Measuring Presets at 8.0, 16.0, 32.0, and 64.0 GT/s §

§ Figure 8-7 illustrate the methodology for measuring Tx equalization coefficients and presets. For a Tx preset to be measured, the DUT Tx transmits a Compliance Pattern with the corresponding Tx equalization coefficients. The

equalized Compliance Pattern is captured by a real-time oscilloscope and the post-processing software extracts an equalized step response waveform. The DUT Tx also transmits a Compliance Pattern with no Tx equalization. The unequalized Compliance Pattern is captured by the real-time oscilloscope and the post-processing software applies Tx equalization coefficients c_{-2} , c_{-1} , c_0 , and c_{+1} to construct an equalized step response waveform. The Tx preset coefficients are the best fit Tx equalization coefficients c_{-2} , c_{-1} , c_0 , and c_{+1} that minimize the Mean Square Error (MSE) between the measured equalized step response waveform and the reconstructed equalized step response waveform.

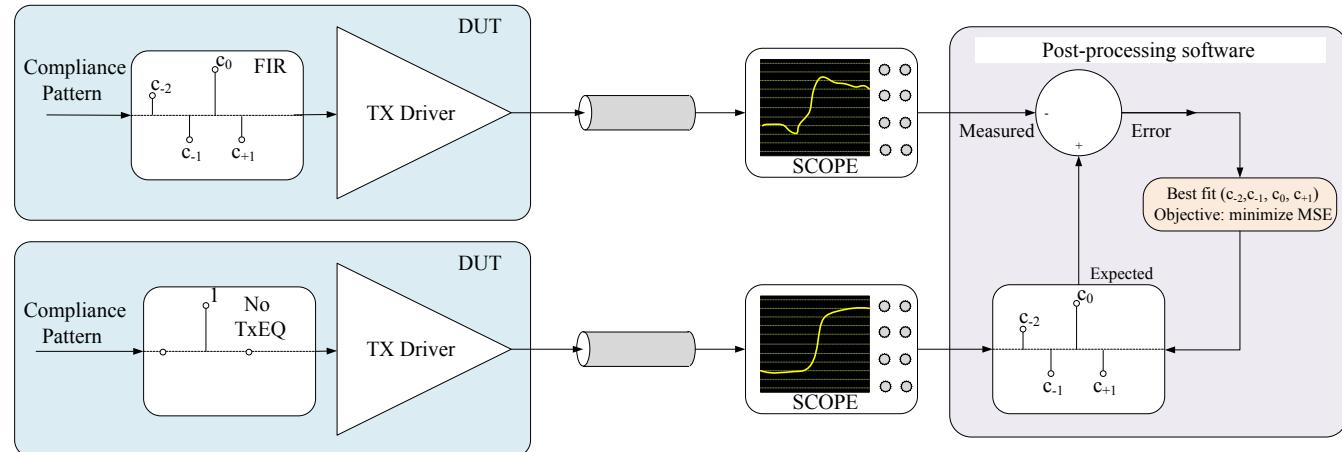


Figure 8-7 Methodology for measuring Tx equalization coefficients and presets §

8.3.3.6 Method for Measuring $V_{TX-DIFF-PP}$ at 2.5 GT/s and 5.0 GT/s §

$V_{TX-DIFF-PP}$ ($V_{TX-DIFF-PP-LOW}$ for reduced swing) at 2.5 GT/s and 5.0 GT/s are measured using the average transition eye amplitude at the 0.5 UI location using 500 repetitions of the compliance pattern.

8.3.3.7 Method for Measuring $V_{TX-DIFF-PP}$ at 8.0, 16.0, 32.0, and 64.0 GT/s §

The range for a Transmitter's output voltage swing, (specified by V_d) with no equalization is defined by $V_{TX-DIFF-PP}$ ($V_{TX-DIFF-PP-LOW}$ for reduced swing), and is obtained by setting c_{-2} , c_{-1} and c_{+1} to zero and measuring the peak-peak voltage on the 64-ones (64 PAM4 voltage level 3's at 64.0 GT/s)/64-zeros (64 PAM4 voltage level 0's at 64.0 GT/s) segment of the compliance pattern. The resulting signal effectively measures at the die pad, minus any low frequency package loss. ISI and switching effects are minimized by restricting the portion of the curve over which voltage is measured to the last few UI of each half cycle, as illustrated in § Figure 8-8 . High frequency noise is mitigated by averaging over 500 repetitions of the compliance pattern.

Base 6.4 vs Base 6.3

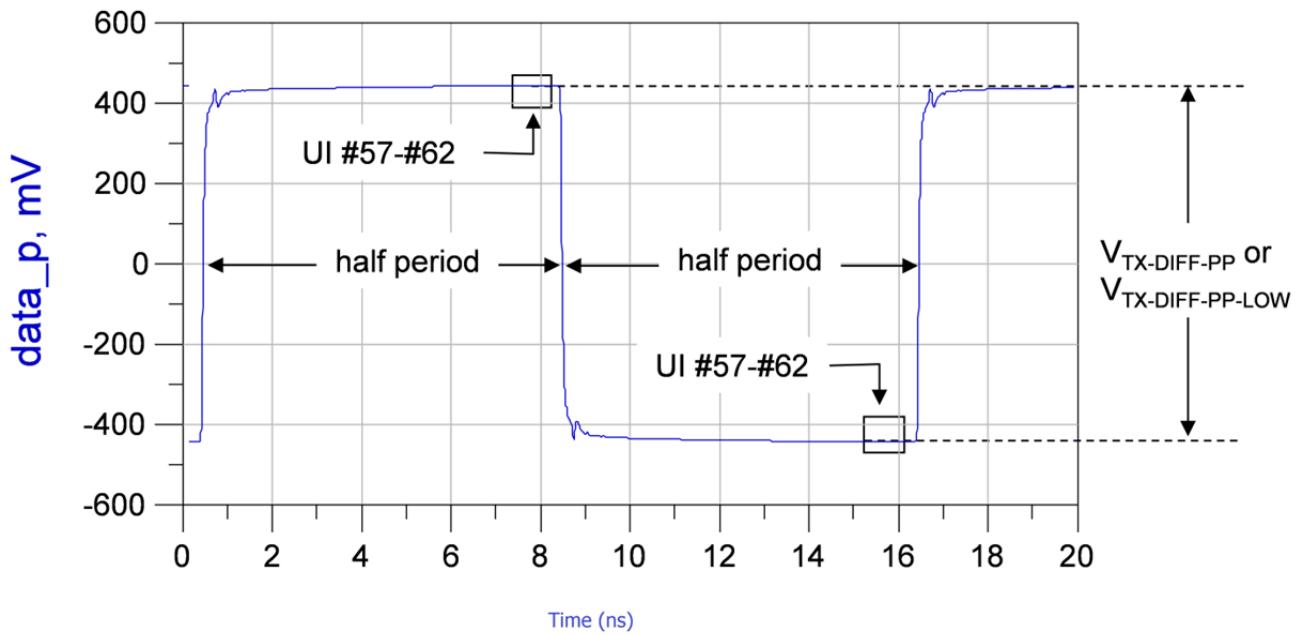


Figure 8-8 $V_{TX-DIFF-PP}$ and $V_{TX-DIFF-PP-LOW}$ Measurement §

8.3.3.8 Coefficient Range and Tolerance for 8.0, 16.0, 32.0, and 64.0 GT/s §

8.0, 16.0, 32.0, and 64.0 GT/s Transmitters are required to inform the Receiver of their coefficient range and tolerance. Coefficient range and tolerance are constrained by the following requirements.

- Coefficients must support all eleven presets and their respective tolerances as defined in § Table 8-1 and § Table 8-2
- All Transmitters must meet the full swing signaling $V_{TX-EIEOS-FS}$ limits.
- Transmitters may optionally support reduced swing, and if they do, they must meet the $V_{TX-EIEOS-RS}$ limits.
- The coefficients must meet the boost and resolution ($V_{TX-BOOST-FS}$, $V_{TX-BOOST-RS}$ and $EQ_{TX-COEFF-RES}$) limits defined in § Table 8-6 .

When the above constraints are applied the resulting coefficient space for 8.0, 16.0, and 32.0 GT/s with pre-shoot2 coefficient $c_{-2} = 0$ may be mapped onto a triangular matrix, an example of which is shown in § Figure 8-9 . The matrix may be interpreted as follows: pre-shoot1 and de-emphasis coefficients are mapped onto the Y-axis and X-axes, respectively. In both cases the maximum granularity of 1/24 is assumed. Each matrix cell, corresponding to a valid combination of pre-shoot1 and de-emphasis coefficients, has three entries corresponding to pre-shoot1 (PS1), de-emphasis (DE), and boost (as shown in the upper left-hand corner). Diagonal elements are defined by the maximum boost ratio. Those cells highlighted in blue are presets required for reduced swing, while cells in either blue or orange represent presets required for full swing signaling. Note that this figure is informative only and is not intended to imply any specific Tx implementation or to alter requirements for nominal preset equalization values and allowed ranges.

Base 6.4 vs Base 6.3

| | | Min Reduced Swing Limit | | | | | | | | |
|-----------------|-----------------|--|-----------------|-----------------|---------------------------|---------------------------|---------------------------|---------------------------|-----------------|----------|
| | | 2 nd Pre-Cursor C ₋₂ = 0/24 (PS2 = 0 dB) | | | | | | | | |
| PS1 DE BOOST | C ₊₁ | | | | | | | | | |
| | | 0/24 | 1/24 | 2/24 | 3/24 | 4/24 | 5/24 | 6/24 | 7/24 | 8/24 |
| C ₋₁ | 0/24 | 0.0 0.0 P4 0.0 | 0.0 -0.8 0.8 | 0.0 -1.6 1.6 | 0.0 -2.5 P3 2.5 | 0.0 -3.5 P1 3.5 | 0.0 -4.7 P2 4.7 | 0.0 -6.0 P0 6.0 | 0.0 -7.6 7.6 | 0.0 -9.5 |
| | 1/24 | 0.8 0.0 0.8 | 0.8 -0.8 1.6 | 0.9 -1.7 2.5 | 1.0 -2.8 3.5 | 1.2 -3.9 4.7 | 1.3 -5.3 6.0 | 1.6 -6.8 7.6 | 1.9 -8.8 9.5 | |
| | 2/24 | 1.6 0.0 P5 1.6 | 1.7 -0.9 2.5 | 1.9 -1.9 3.5 | 2.2 -3.1 4.7 | 2.5 -4.4 6.0 | 2.9 -6.0 P7 7.6 | 3.5 -8.0 9.5 | | |
| | 3/24 | 2.5 0.0 P6 2.5 | 2.8 -1.0 3.5 | 3.1 -2.2 4.7 | 3.5 -3.5 P8 6.0 | 4.1 -5.1 7.6 | 4.9 -7.0 9.5 | | | |
| | 4/24 | 3.5 0.0 P9 3.5 | 3.9 -1.2 4.7 | 4.4 -2.5 6.0 | 5.1 -4.1 7.6 | 6.0 -6.0 9.5 | | | | |
| | 5/24 | 4.7 0.0 4.7 | 5.3 -1.3 6.0 | 6.0 -2.9 7.6 | 7.0 -4.9 9.5 | | | | | |
| | 6/24 | 6.0 0.0 6.0 | 6.8 -1.6 7.6 | 8.0 -3.5 9.5 | | | | | | |

Full Swing Limit or
Max Reduced Swing Limit

Figure 8-9 Transmit Equalization Coefficient Space Triangular Matrix Example for 8.0, 16.0, and 32.0 GT/s §

The coefficient space for 64.0 GT/s with each pre-shoot2 coefficient may be mapped onto a triangular matrix, an example of which is shown in § Figure 8-10 . Maximum granularity of 1/24 is assumed for Tx equalization coefficient. Those cells highlighted in blue are presets required for reduced swing, while cells in either blue or orange represent presets required for full swing signaling. Note that this figure is informative only and is not intended to imply any specific Tx implementation or to alter requirements for nominal preset equalization values and allowed ranges.

Base 6.4 vs Base 6.3

| | | 2 nd Pre-Cursor C ₂ = 0/24 | | | | | | | | | | | |
|--|------|--|-----|------|------|------|------|------|------|------|------|------|------|
| PRESET | PS2 | PS1 | DE | C+1 | | | | | | | | | |
| | | | | 0/24 | 1/24 | 2/24 | 3/24 | 4/24 | 5/24 | 6/24 | 7/24 | 8/24 | |
| C_1 | 0/24 | 0.0 | 0.0 | 0.0 | 0.0 | -0.8 | 0.0 | 0.0 | -1.6 | 0.0 | 0.0 | -3.5 | |
| | | Q0 | 0.0 | 0.8 | Q3 | 1.6 | 0.0 | 0.0 | -2.5 | Q4 | 3.5 | 0.0 | 0.0 |
| | 1/24 | 0.0 | 0.8 | 0.0 | 0.0 | -0.8 | 0.0 | 0.9 | -1.7 | 0.0 | 1.2 | -3.9 | |
| | | 0.8 | 1.6 | 2.5 | 3.5 | 0.0 | 1.0 | -2.8 | 4.7 | 0.0 | 1.3 | -5.3 | |
| | 2/24 | 0.0 | 1.6 | 0.0 | 0.0 | 1.7 | -0.9 | 0.0 | 1.9 | -1.9 | 0.0 | 2.2 | -3.1 |
| | | Q1 | 1.6 | 2.5 | 3.5 | 4.7 | 0.0 | 2.2 | -3.1 | 6.0 | 0.0 | 2.5 | -4.4 |
| | 3/24 | 0.0 | 2.5 | 0.0 | 0.0 | 2.8 | -1.0 | 0.0 | 3.1 | -2.2 | 0.0 | 3.5 | -3.5 |
| | | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 0.0 | 3.5 | -3.5 | 7.6 | 0.0 | 4.9 | -7.0 |
| C_1 | 4/24 | 0.0 | 3.5 | 0.0 | 0.0 | 3.9 | -1.2 | 0.0 | 4.4 | -2.5 | 0.0 | 5.1 | -4.1 |
| | | Q2 | 3.5 | 4.7 | 6.0 | 7.6 | 0.0 | 4.4 | -2.5 | 9.5 | 0.0 | 6.0 | -6.0 |
| | 5/24 | 0.0 | 4.7 | 0.0 | 0.0 | 5.3 | -1.3 | 0.0 | 6.0 | -2.9 | 0.0 | 7.0 | -4.9 |
| | | 4.7 | 6.0 | 7.6 | 9.5 | 0.0 | 6.0 | -2.9 | 9.5 | 0.0 | 7.0 | -4.9 | |
| | 6/24 | 0.0 | 6.0 | 0.0 | 0.0 | 6.9 | -1.6 | 0.0 | 8.0 | -3.5 | 0.0 | 9.5 | -9.5 |
| | | 6.0 | 7.6 | 9.5 | 0.0 | 6.9 | -1.6 | 0.0 | 8.0 | -3.5 | 0.0 | 9.5 | -9.5 |
| Full Swing Limit or Max Reduced Swing Limit | | | | | | | | | | | | | |
| Min Reduced Swing Limit | | | | | | | | | | | | | |
| | | 2 nd Pre-Cursor C ₂ = 1/24 | | | | | | | | | | | |
| PRESET | PS2 | PS1 | DE | C+1 | | | | | | | | | |
| | | | | 0/24 | 1/24 | 2/24 | 3/24 | 4/24 | 5/24 | 6/24 | 7/24 | 8/24 | |
| C_1 | 0/24 | -0.8 | 0.0 | 0.0 | -0.8 | 0.0 | -0.8 | -0.9 | 0.0 | -1.6 | -1.0 | 0.0 | -2.5 |
| | | 0.0 | 0.8 | 1.6 | 2.5 | 3.5 | 4.7 | 5.6 | 6.0 | 7.6 | -1.9 | 0.0 | -7.6 |
| | 1/24 | -0.8 | 0.8 | 0.0 | -0.9 | 0.8 | -0.8 | -1.0 | 0.9 | -1.7 | -1.2 | 1.0 | -2.8 |
| | | 0.8 | 1.6 | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 8.0 | 9.5 | -1.9 | 1.6 | -6.8 |
| | 2/24 | -0.9 | 1.6 | 0.0 | -1.0 | 1.7 | -0.9 | -1.2 | 1.9 | -1.9 | -1.3 | 1.2 | -3.9 |
| | | 1.6 | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 8.0 | 9.5 | 0.0 | 2.5 | -4.4 | |
| | 3/24 | -1.0 | 2.5 | 0.0 | -1.2 | 2.8 | -1.0 | -1.3 | 3.1 | -2.2 | -1.6 | 2.2 | -3.1 |
| | | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 8.0 | 9.5 | 0.0 | 2.5 | -4.4 | | |
| C_1 | 4/24 | -1.2 | 3.5 | 0.0 | -1.3 | 3.9 | -1.2 | -1.6 | 4.4 | -2.5 | -1.9 | 4.1 | -5.1 |
| | | 3.5 | 4.7 | 6.0 | 7.6 | 9.5 | 0.0 | 4.4 | -2.5 | 9.5 | -2.5 | 4.9 | -7.0 |
| | 5/24 | -1.3 | 4.7 | 0.0 | -1.6 | 5.3 | -1.3 | -1.9 | 6.0 | -2.9 | -2.5 | 7.0 | -4.9 |
| | | 4.7 | 6.0 | 7.6 | 9.5 | 0.0 | 6.0 | -2.9 | 9.5 | 0.0 | 7.0 | -4.9 | |
| | 6/24 | -1.6 | 6.0 | 0.0 | -1.9 | 6.9 | -1.6 | -2.5 | 8.0 | -3.5 | -1.6 | 6.0 | -6.0 |
| | | 6.0 | 7.6 | 9.5 | 0.0 | 6.9 | -1.6 | 0.0 | 8.0 | -3.5 | 0.0 | 9.5 | -9.5 |
| Full Swing Limit or Max Reduced Swing Limit | | | | | | | | | | | | | |
| Min Reduced Swing Limit | | | | | | | | | | | | | |
| | | 2 nd Pre-Cursor C ₂ = 2/24 | | | | | | | | | | | |
| PRESET | PS2 | PS1 | DE | C+1 | | | | | | | | | |
| | | | | 0/24 | 1/24 | 2/24 | 3/24 | 4/24 | 5/24 | 6/24 | 7/24 | 8/24 | |
| C_1 | 0/24 | -1.6 | 0.0 | 0.0 | -1.7 | 0.0 | -0.8 | -1.9 | 0.0 | -1.6 | -2.2 | 0.0 | -3.5 |
| | | 0.0 | 0.8 | 1.6 | 2.5 | 3.5 | 4.7 | 5.6 | 6.0 | 7.6 | -4.4 | 0.0 | -7.6 |
| | 1/24 | -1.7 | 0.8 | 0.0 | -1.9 | 0.8 | -0.8 | -2.2 | 0.9 | -1.7 | -2.5 | 1.0 | -2.8 |
| | | 0.8 | 1.6 | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 8.0 | 9.5 | -4.4 | 1.6 | -6.8 |
| | 2/24 | -1.9 | 1.6 | 0.0 | -2.2 | 1.7 | -0.9 | -2.5 | 1.9 | -1.9 | -2.9 | 1.2 | -3.9 |
| | | 1.6 | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 8.0 | 9.5 | 0.0 | 2.5 | -4.4 | |
| | 3/24 | -2.2 | 2.5 | 0.0 | -2.5 | 2.8 | -1.0 | -2.9 | 3.1 | -2.2 | -3.5 | 2.5 | -4.4 |
| | | 2.5 | 3.5 | 4.7 | 6.0 | 7.6 | 8.0 | 9.5 | 0.0 | 2.5 | -4.4 | | |
| C_1 | 4/24 | -2.5 | 3.5 | 0.0 | -2.9 | 3.9 | -1.2 | -3.5 | 4.4 | -2.5 | -4.4 | 5.1 | -4.1 |
| | | 3.5 | 4.7 | 6.0 | 7.6 | 9.5 | 0.0 | 4.4 | -2.5 | 9.5 | -6.0 | 6.0 | -6.0 |
| | 5/24 | -2.9 | 4.7 | 0.0 | -3.5 | 5.3 | -1.3 | -4.4 | 6.0 | -2.9 | -4.4 | 7.0 | -4.9 |
| | | 4.7 | 6.0 | 7.6 | 9.5 | 0.0 | 6.0 | -2.9 | 9.5 | 0.0 | 7.0 | -4.9 | |
| | 6/24 | -3.5 | 6.0 | 0.0 | -4.4 | 6.9 | -1.6 | -6.0 | 8.0 | -3.5 | -4.4 | 9.6 | -9.5 |
| | | 6.0 | 7.6 | 9.5 | 0.0 | 6.9 | -1.6 | 0.0 | 8.0 | -3.5 | 0.0 | 9.5 | -9.5 |
| Full Swing Limit or Max Reduced Swing Limit | | | | | | | | | | | | | |
| Min Reduced Swing Limit | | | | | | | | | | | | | |

Figure 8-10 Transmit Equalization Coefficient Space Triangular Matrix Example for 64.0 GT/s §

8.3.3.9 EIEOS and $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ Limits §

EIEOS signaling is defined for 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s only.

- At 5.0 GT/s the K28.7 Symbol is used to exit Electrical Idle.
- At 8.0 GT/s the EIEOS pattern consists of 8 consecutive zeros followed by the same number of consecutive ones, where the pattern is repeated for a total of 128 UI.
- At 16.0 GT/s the EIEOS pattern consists of 16 consecutive zeros followed by the same number of consecutive ones, where the pattern is repeated for a total of 128 UI.
- At 32.0 GT/s the EIEOS pattern consists of 32 consecutive zeros followed by the same number of consecutive ones, where the pattern is repeated for a total of 128 UI.
- At 64.0 GT/s the EIEOS pattern consists of 32 UI consecutive voltage level 0's followed by 32 UI consecutive voltage level 3's (see § Section 4.2.5.3).

A transmitter sends an EIEOS to cause an exit of Electrical Idle at the Receiver. This pattern guarantees the Receiver will properly detect the EI Exit condition with its squelch exit detect circuit, something not otherwise guaranteed by scrambled data. The Tx EIEOS launch voltage is defined by $V_{TX-EIEOS-FS}$ for full swing signaling and by $V_{TX-EIEOS-RS}$ for reduced swing signaling. $V_{TX-EIEOS-RS}$ is smaller than $V_{TX-EIEOS-FS}$ to reflect the fact that reduced swing is typically supported only for lower loss channels where there is less attenuation at the EIEOS signaling rate.

$V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ are measured using the EIEOS sequence contained within the compliance pattern for 8.0, 16.0, 32.0, and 64.0 GT/s.

For full swing signaling $V_{TX-EIEOS-FS}$ is measured with a preset number P10 for 8.0, 16.0, and 32.0 GT/s, and with a preset number Q10 for 64.0 GT/s. This is equivalent to a maximum nominal boost of 9.5 dB and represents the maximum boost attainable in coefficient space. When a tolerance of ± 1.5 dB is factored in this yields the minimum boost limit of 8.0 dB appearing in § Table 8-6. For reduced swing signaling $V_{TX-EIEOS-RS}$ is measured with preset P1 for 8.0, 16.0, and 32.0 GT/s, and with a preset number Q4 for 64.0 GT/s.

A Transmitter is not always permitted to generate the maximum boost level noted above. A Transmitter that cannot drive significantly more than 800 mVPP is limited by the need to meet $V_{TX-EIEOS-FS}$. The Tx must reject any adjustments to its presets or coefficients that would violate the $V_{TX-EIEOS-FS}$ or $V_{TX-EIEOS-RS}$ limits. The EIEOS voltage limits are imposed to guarantee the EIEOS threshold of 175 mVPP at the Rx pin.

§ Figure 8-11 illustrates the de-emphasis peak as observed at the pin of a Tx for $V_{TX-EIEOS-FS}$. At the far end of a lossy channel the de-emphasis peak will be attenuated; this is why the measurement interval includes only the middle five UI at 8.0 GT/s, UI number 5-14 at 16.0 GT/s, and UI number 9-28 at 32.0 and 64.0 GT/s. The voltage is averaged over this interval for both the negative and positive halves of the waveform over 500 repetitions of the compliance pattern.

$V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ are defined as the difference between the negative and positive waveform segment averages. UI boundaries are defined with respect to the edge of the recovered data clock.

Base 6.4 vs Base 6.3

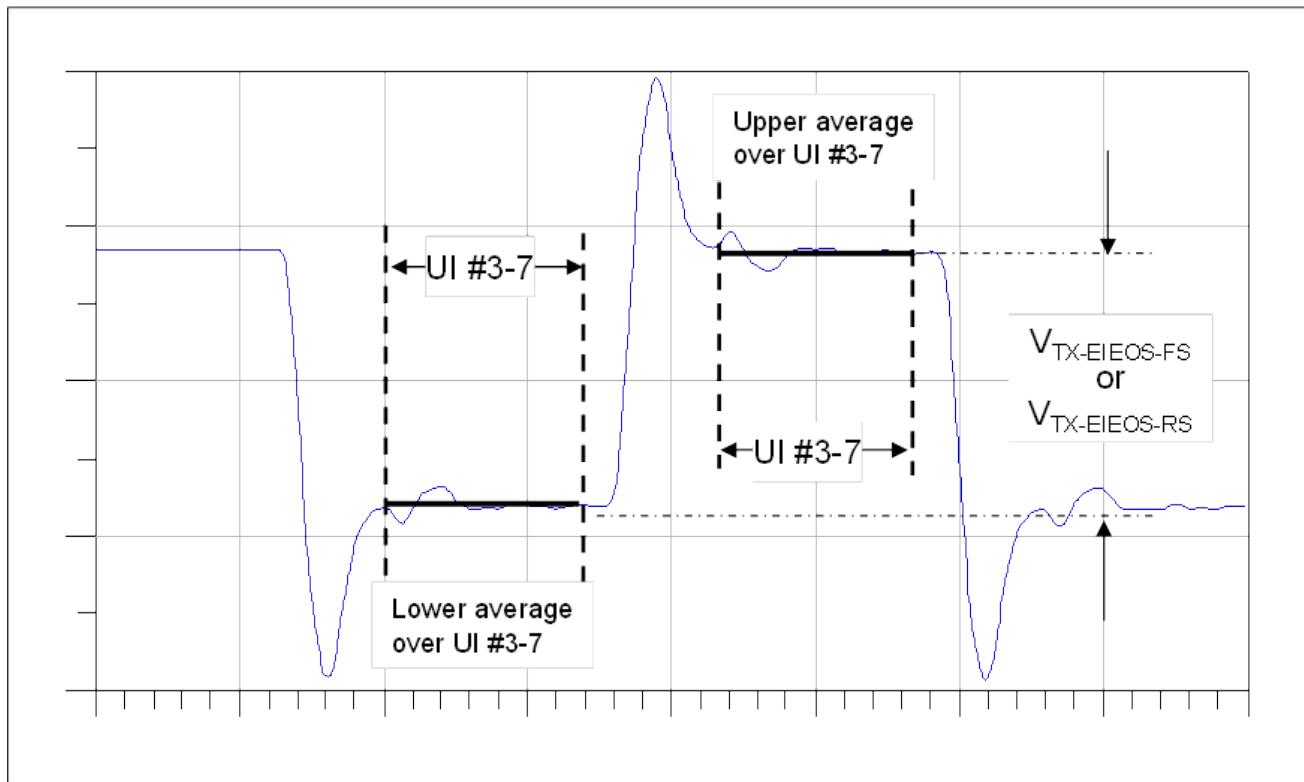


Figure 8-11 Measuring $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ at 8.0 GT/s §

8.3.3.10 Reduced Swing Signaling §

PCI Express Transmitters may optionally support a reduced swing signaling. It is left as an implementation option to define the maximum reduced swing voltage value for $V_{TX-DIFF-PP-LOW}$ anywhere up to the maximum full swing voltage. The minimum for $V_{TX-DIFF-PP-LOW}$ is captured indirectly by the constraint imposed by $V_{TX-EIEOS-RS}$, so there is no need to define a separate minimum limit for $V_{TX-DIFF-PP-LOW}$. Reduced swing limits the range of presets and the maximum boost. The boost for reduced swing must be in the region shown in § Figure 8-9 and § Figure 8-10 between the Max reduced swing limit and minimum reduced swing boost limit.

Form factors are permitted to disallow, optionally allow, or require Reduced Swing Signaling, depending on the channel requirements for the form factor. When Reduced Swing Signaling is allowed or required it is required that form factor specifications provide any additional details necessary to support interoperability.

8.3.3.11 Effective Tx Package Loss at 8.0, 16.0, 32.0, and 64.0 GT/s §

Package loss (including silicon driver bandwidth) is represented by the $ps21_{TX}$ parameter. Since both package IL and driver bandwidth affect the signal as observed at the Tx pin, the $ps21_{TX}$ parameter has the advantage of representing both of these effects, while permitting the measurement to be made at a point (TP1) that can easily be probed. It is necessary to include a package loss parameter in the Tx specification, since the voltage swing parameters ($V_{TX-DIFF-PP}$ and $V_{TX-DIFF-PP-LOW}$) are defined at an equivalent pulse frequency of 1/128 UI and purposely do not capture high frequency driver or package loss effects.

At 16.0 GT/s and 32.0 GT/s, separate ps21_{TX} parameters are defined for packages containing Root Ports (Root Package) and for all other packages (Non-Root Package), based on the assumption that the former tend to be large and require socketing, while the latter are smaller and usually not socketed.

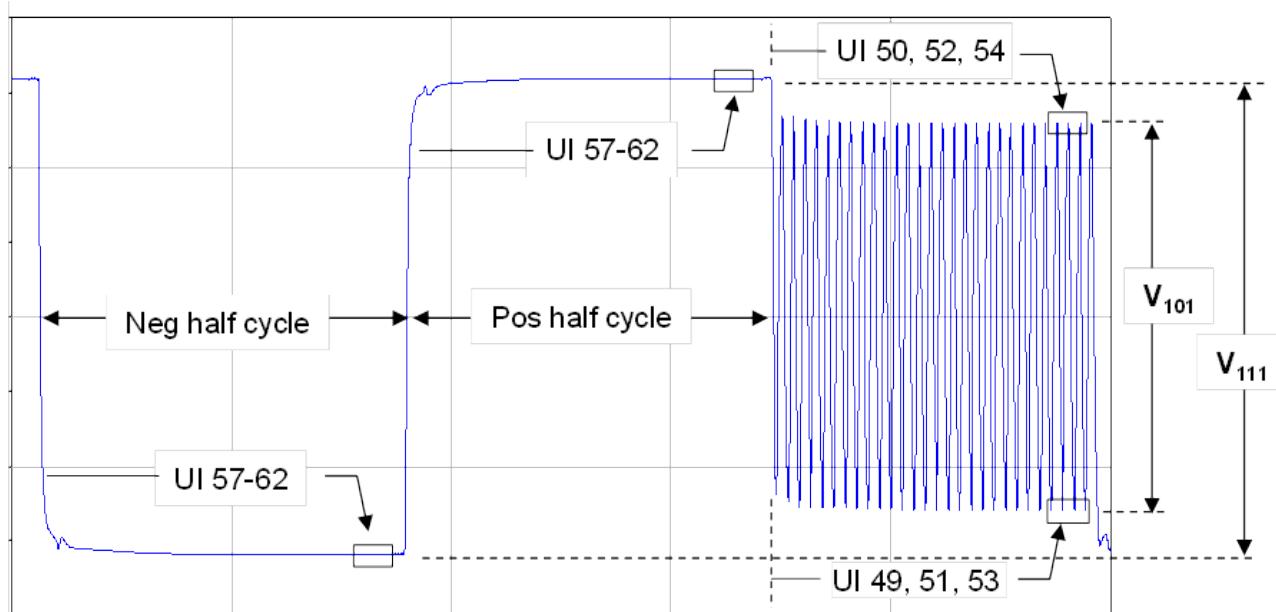
The ps21_{TX} parameter is informative for 16.0 GT/s Root Package devices and for Non-Root Package devices that only support PCI Express standard form factors (i.e., CEM, M.2, etc.). The ps21_{TX} parameter is normative for all 8.0, 32.0, and 64.0 GT/s devices and for 16.0 GT/s Non-Root Package devices that support captive channels. (See § Table 8-3 below).

Table 8-3 Cases that the Reference Packages and ps21_{TX} Parameter are Normative §

| | 8.0, 32.0, and 64.0 GT/s | | 16.0 GT/s | |
|--|--------------------------|-------------------------|---------------------|-------------------------|
| | Root Package Device | Non-Root Package Device | Root Package Device | Non-Root Package Device |
| Device supports captive channels | Normative | Normative | Informative | Normative |
| Device does not support captive channels | Normative | Normative | Informative | Informative |

All implementations of PCI Express standard form factors must still meet form factor requirements. Devices for which the ps21_{TX} parameter is informative, as defined above must provide a package model for use in channel compliance if they do not meet the informative ps21_{TX} parameter.

Package loss is measured by comparing the 64-zeros/64-ones voltage swing (V_{111}) against a 1010 pattern (V_{101}). Tx package loss measurement is made with c_{-2} , c_{-1} , and c_{+1} set to zero. Measurements shall be made averaging over 500 repetitions of the compliance pattern.



$$\text{ps21}_{\text{TX}} = 20 \log_{10} (V_{101}/V_{111})$$

Figure 8-12 Compliance Pattern and Resulting Package Loss Test Waveform §

Measurement of V_{101} and V_{111} is made towards the end of each interval to minimize ISI and low frequency effects. V_{101} is defined as the peak-peak voltage between minima and maxima of the clock pattern. V_{111} is defined as the average voltage difference between the positive and negative levels of the two half cycles. The measurement should be averaged over 500 repetitions of the compliance pattern.

At 32.0 GT/s only the $ps21_{TX}$ parameter is calculated by filtering the captured voltage waveforms normally used for $ps21_{TX}$ measurements as follows:

- V_{111} is measured from a filtered voltage waveform with a first order (20 dB/decade) low pass filter with a -3 dB corner frequency at 1 GHz applied.
- V_{101} is measured from a filtered voltage waveform with a first order (20 dB/decade) high pass filter with a -3 dB corner frequency at 7 GHz ↑ and a low-pass filter with zero phase, no more than 0.1 dB attenuation at 16 GHz, no less than 40 dB of attenuation at frequencies above 48 GHz and cutoff frequency between 32 GHz to 40 GHz ↑ applied. ↑ If the low-pass filter causes the $ps21_{TX}$ parameter to fail the specification, then V_{101} can be remeasured without the low-pass filter. ↑

ECN: Base 6.3
Tx-PS21△↓

Since the Nyquist frequency for 64.0 GT/s PAM4 signaling is 16.0 GHz, no additional measurement is necessary for 64.0 GT/s $ps21_{TX}$ spec compliance. For a PCIe 6.0 device that supports maximum data rate of 64.0 GT/s, the $ps21_{TX}$ spec parameter measured for 32.0 GT/s must be lower than the $ps21_{TX}$ spec values provided under the 64.0 GT/s column in § Table 8-6 .

8.3.3.12 Linear Fit Pulse Response §

Several PCIe measurements are based on a Linear Fit Pulse Response (LFPR) model. This section provides a description of the general method. The basic LFPR is a linear model $y(k) = x(n) * p$, where '*' denotes convolution, $x(n)$ is an input stream of test pattern symbols where symbols {0, 1, 2, 3} are represented using {-1, -1/3, 1/3, 1} respectively, $y(k)$ is an output stream of waveform samples, and p is the linear fit pulse response which maps ideal symbols to measured samples.

A waveform is captured from a DUT (device under test) with an effective sample rate that is M times the signaling rate of the DUT, where M is an integer, and resampling may be used to meet this requirement. The waveform is then averaged using multiple pattern repetitions. Thus, if the pattern length is N and the number of pattern repetitions is R_{PAT} , then $R_{PAT} \cdot N \cdot M$ samples are averaged into a single pattern repetition of length $N \cdot M$ samples. The samples of this waveform are denoted as $y(k)$. The symbols $x(n)$ are aligned with $y(k)$ such that the first M samples of $y(k)$ correspond to the first symbol of the test pattern $x(n)$, the second M samples of $y(k)$ to the second symbol, and so on. A linear system of equations is solved for vector p . An efficient way to solve the system is to organize it as the matrix equation $PX = Y$. The M -by- N waveform matrix Y is defined as:

$$Y = \begin{bmatrix} y(1) & y(M+1) & \vdots & y(M(N-1)+1) \\ y(2) & y(M+2) & \vdots & y(M(N-1)+2) \\ \vdots & \vdots & \vdots & \vdots \\ y(M) & y(2M) & \vdots & y(MN) \end{bmatrix}$$

Equation 8-3 Y §

Rotate the symbols vector x by a specified pulse delay D_p to yield x_r as shown in § Equation 8-4 .

$$x_r = \begin{bmatrix} x(D_p + 1) & x(D_p + 2) & \vdots & x(N) & x(1) & \vdots & x(D_p) \end{bmatrix}$$

[Equation 8-4 \$x_r\$](#) §

Define the matrix X to be an N -by- N matrix derived from x_r , as shown in § Equation 8-5 .

$$X = \begin{bmatrix} x_r(1) & x_r(2) & \vdots & x_r(N) \\ x_r(N) & x_r(1) & \vdots & x_r(N - 1) \\ \vdots & \vdots & \vdots & \vdots \\ x_r(2) & x_r(3) & \vdots & x_r(1) \end{bmatrix}$$

[Equation 8-5 \$X\$](#) §

Define the matrix X_1 to be the first N_p rows of X concatenated with a row vector of ones of length N , where N_p is a specified pulse length. The M -by-($N_p + 1$) coefficient matrix, P , corresponding to the linear fit pulse response is then defined by § Equation 8-6 . The superscript “ T ” denotes the matrix transpose operator.

$$P = Y X_1^T (X_1 X_1^T)^{-1}$$

[Equation 8-6 \$P\$](#) §

The pulse fit vector $p(k)$ is read column-wise from the first N_p columns of P . The error waveform, $e(k)$, is read column-wise from the elements of E as shown in § Equation 8-7

$$E = P X_1 - Y = \begin{bmatrix} e(1) & e(M + 1) & \vdots & e(M(N - 1) + 1) \\ e(2) & e(M + 2) & \vdots & e(M(N - 1) + 2) \\ \vdots & \vdots & \vdots & \vdots \\ e(M) & e(2M) & \vdots & e(MN) \end{bmatrix}$$

[Equation 8-7 \$E\$](#) §

The distortion term, also called fitting error, σ_e , is defined as the root mean squared (RMS) value of the error waveform $e(k)$.

8.3.3.13 Transmitter Signal-to Noise and Distortion Ratio (SNDR_{TX}) for 64.0 GT/s §

Signal-to-noise and distortion ratio (SNDR) is measured at the transmitter output using the Compliance Pattern (see § Section 4.2.14) with preset Q₀ (no Tx equalization), and the lanes not under test also transmitting the Compliance Pattern with preset Q₀. The recorded waveform must have a minimum of 250 repetitions (R_{PAT}) of the compliance pattern. Measurements should be made with a 4th order Bessel-Thomson filter with a roll-off from DC value by 3 dB at 33 GHz to minimize the impact of scope high-frequency noise. The minimum scope bandwidth is 50 GHz.

§ Section 8.3.3.12 describes the Linear Fit Pulse Response method where a single pulse response is created from an averaged transmitter waveform. The Multi Pulse Response Fit (MPRF) method is introduced as the terminology for creating more than one linear fit pulse responses. For the SNDR_{TX} measurement, two pulse responses (even and odd) are used to mitigate the impact of duty cycle errors. Another variant of the MPRF will be used in § Section 8.3.3.14 with four pulse responses, one for each PAM4 signal level. Let x be the input stream of transmitted symbols and y be the output stream of captured waveform. In this method, two pulse responses are defined, denoted as p_E and p_O, to provide the best possible approximation to § Equation 8-8.

$$y = p_E * (x \cdot \mathbf{1}_E) + p_O * (x \cdot \mathbf{1}_O)$$

Equation 8-8 y §

where '*' represents convolution. Here, $\mathbf{1}_E$ is an indicator function, or mask, with a value of 1 for even UIs and 0 for odd UIs, while $\mathbf{1}_O$ has a value of 1 for odd UIs and 0 for even UIs. The notations $(x \cdot \mathbf{1}_E)$ and $(x \cdot \mathbf{1}_O)$ define pointwise multiplication, where each element in x is multiplied by its corresponding element in either $\mathbf{1}_E$ or $\mathbf{1}_O$.

To solve the system, define the matrix X_{NP} to be the first N_p rows of X from § Equation 8-5 and construct the (2 N_p + 1)-by- N block linear system

$$X_{EO} = \begin{bmatrix} X_{NP} \cdot \mathbf{1}_E \\ X_{NP} \cdot \mathbf{1}_O \\ 1 1 \dots 1 \end{bmatrix}$$

Equation 8-9 X_{NP} §

where masks $\mathbf{1}_E$ and $\mathbf{1}_O$ are organized here as a 2D array (but play the same role), and 1 1 ... 1 is a row of ones. Next, solve the linear system by computing,

$$P_{EO} = Y X_{EO}^T (X_{EO} X_{EO}^T)^{-1}$$

Equation 8-10 P_{EO} §

with Y as defined in § Equation 8-3. The pulse fit vector p_E is read column-wise from the first N_p columns of P_{EO}, while p_O is read column-wise from the next N_p columns.

The pulse responses p_E and p_O each have a length of N_p UI, and when oversampling by a factor of M (the number of samples per PAM4 symbol), the array sizes for p_E , p_O , and e become $N_p M$. For PCIe specific case, N_p is set to 600, the pulse delay D_p is 4, and M should be a minimum of 32.

The error vector, e , is defined by § [Equation 8-11](#)

$$e = p_E * (x \cdot \mathbf{1}_E) + p_O * (x \cdot \mathbf{1}_O) - y$$

[Equation 8-11 e](#) §

and can be computed using P_{EO} and X_{EO} in § [Equation 8-10](#). The standard deviation of the error vector, denoted as σ_e , is obtained from the measured PRBS portion of the compliance pattern.

The parameter σ_n measures the uncorrelated RMS amplitude noise of each symbol level (including random noise and uncorrelated bounded noise effects), while not including ISI and jitter effects. Noise for each of the four PAM4 voltage levels, σ_L , is measured by using the PAM4 symbol 61 of the 64-UI long slow pattern for the corresponding voltage level that appears once in every repeat of the Compliance Pattern. When measuring σ_L , an adjustment for uncorrelated random noise contributed by the instrumentation such as uncorrelated random scope noise shall be applied. Equivalent oscilloscope settings used for noise characterization shall be consistent with the oscilloscope settings used for waveform capture when calculating SNDR_{TX}. For each voltage level L (where $L = 0,1,2,3$), the σ_L measurement is the result of eight independent measurements on eight evenly spaced sample points within the Unit Interval of symbol 61 in the run of 64 identical symbols. Each of the eight measurements denoted as $\sigma_{L,i}$ (where $i=1..8$) is calculated by using the following equations.

$$\sigma_{L,i}^2 = \frac{1}{N_k} \sum_{k=1}^{N_k} (V_{L,i,k} - \mu_{L,i})^2$$

[Equation 8-12](#) $\sigma_{L,i}$ §

$$\mu_{L,i} = \frac{1}{N_k} \sum_{k=1}^{N_k} V_{L,i,k}$$

[Equation 8-13](#) $\mu_{L,i}$ §

In the above equations, N_k is the number of repetitions of the compliance pattern in the recorded waveform, $V_{L,i,k}$ is the waveform voltage at the i th data sample location within the 61st symbol UI in the k th repetition of the compliance pattern for each voltage level, and $\mu_{L,i}$ is the mean of N_k waveform voltage samples for the i th data sample location for the corresponding voltage level.

σ_L is obtained via the following equation:

$$\sigma_L = \sqrt{\frac{\sum_{i=1}^8 \sigma_{L,i}^2}{8}}$$

Equation 8-14 σ_L §

σ_n is the average of the four σ_L measurements, one for each PAM4 voltage level, denoted as

$$\sigma_n = \frac{1}{4}(\sigma_0 + \sigma_1 + \sigma_2 + \sigma_3)$$

Equation 8-15 σ_n §

The Tx SNDR_{TX} is defined by § [Equation 8-16](#)

$$SNDR = 10 \times \log_{10} \left| \frac{\sum_{k \in S_E} P_E(k)^2 + \sum_{k \in S_O} P_O(k)^2}{2(\sigma_e^2 + \sigma_n^2)} \right|$$

Equation 8-16 SNDR §

Sets S_E and S_O are associated with pulse response p_E and p_O respectively. To obtain set S from a pulse response p , follow these steps:

1. Find the index of pulse response peak, $j : P(j) = p_{max}$
2. Pick all indexes of p that have the same within-UI offset as the peak index j .
3. Mathematically, set S is defined as:

$$S = \{k \mid k = j + nM, n \in \mathbb{Z} \text{ and } 0 \leq k < N_p M\}$$

Note that j may not be the same for p_E and p_O .

8.3.3.14 Transmitter Ratio of Level Mismatch (R_{LM-TX}) for 64.0 GT/s

Transmitter linearity is defined as a function of the PAM4 signal levels (V_0, V_1, V_2 , and V_3) transmitted for PAM4 2-bit symbols (see § [Section 4.2.3.1.1](#)). The ratio of level mismatch, R_{LM} , is defined as shown below:

$$V_{mid} = \left(V_0 + V_3 \right) / 2$$

$$ES_1 = \left(V_1 - V_{mid} \right) / \left(V_0 - V_{mid} \right)$$

$$ES_2 = \left(V_2 - V_{mid} \right) / \left(V_3 - V_{mid} \right)$$

$$R_{LM} = \min \left(\left| 3 \times ES_1 \right|, \left| 3 \times ES_2 \right|, \left| 2 - 3 \times ES_1 \right|, \left| 2 - 3 \times ES_2 \right| \right)$$

Equation 8-17 R_{LM} §

The PAM4 signal levels (V_L where $L = 0, 1, 2$, and 3) described above are measured by following a variant of the Multi Pulse Response Fit method. The method of obtaining the V_L values described in this section only applies to calculation of R_{LM-TX} .

8.3.3.14.1 Multi Pulse Response Fit (MPRF) – PAM4 Voltage Variant §

To quantify the linearity, the levels of symbols 0 and 3 are used as reference levels. The deviation of levels of symbol 1 and 2 from even spacing between the reference levels is then measured. For the Linear Fit Pulse Response (multiple pulses) method used for PAM4 signaling, three pulses are extracted: one normalized pulse representing both symbol 0 and symbol 3 , one normalized pulse for symbol 1 and one normalized pulse for symbol 2 .

The steps in § Section 8.3.3.12 describing the Linear Fit Pulse Response (single pulse) method will be followed through § Equation 8-5 using M greater than or equal to 32 , at least 250 pattern repeats (R_{PAT}), the pulse delay D_p of 4 , and a pulse length (N_p) of 600 .

Define the matrix X_{NP} to be the first N_p rows of X .

Define the following three N_p -by- N switch matrices for the three normalized pulses as

$$W_{sym0_3} = \left(X_{N_p} == -1 \right) \text{ or } \left(X_{N_p} == 1 \right)$$

$$W_{sym1} = \left(X_{N_p} == -1 \mid 3 \right)$$

$$W_{sym2} = \left(X_{N_p} == 1 \mid 3 \right)$$

Equation 8-18 W_{sym0_3} §

Where the “==” operation returns Boolean value. If the Boolean value in an element in the matrix is true where the “==” condition is satisfied, then the corresponding element in W_{sym} is set to 1. If the Boolean value is false where the “==” condition is not satisfied, the element in W_{sym} is set to 0. These matrices serve the same purpose as the indicator function, or mask, functions in § [Section 8.3.3.13](#).

Denote

$$X_{sym} = \begin{bmatrix} X_{N_p} \cdot W_{sym0_3} \\ X_{N_p} \cdot W_{sym1} \\ X_{N_p} \cdot W_{sym2} \\ 1 1 \dots 1 \end{bmatrix}$$

Equation 8-19 X_{sym} §

The notation ($X_{N_p} \cdot W_{sym^*}$) defines pointwise multiplication, where each element in X_{N_p} is multiplied by its corresponding element in W_{sym^*} . The 1 1 1 ... 1 notation denotes a row vector of ones.

Calculate the following equation to get:

$$P_{sym} = Y X_{sym}^T \left(X_{sym} X_{sym}^T \right)^{-1}$$

Equation 8-20 P_{sym} §

Let P_{sym0_3} be the first N_p columns of P_{sym} , then P_{sym1} be the next N_p columns of P_{sym} , and P_{sym2} be the next N_p columns. The normalized linear fit pulse response for the symbol 0 and symbol 3, $p_{sym0_3}(k)$, is then read column-wise from the elements of P_{sym0_3} . The normalized linear fit pulse response for symbol 1, $p_{sym1}(k)$, is read column-wise from the elements of P_{sym1} . The normalized linear fit pulse response for symbol 2, p_{sym2} , is read column-wise from the elements of P_{sym2} .

For an example waveform, its symbol pulses are obtained as shown in § Figure 8-13 :

Base 6.4 vs Base 6.3

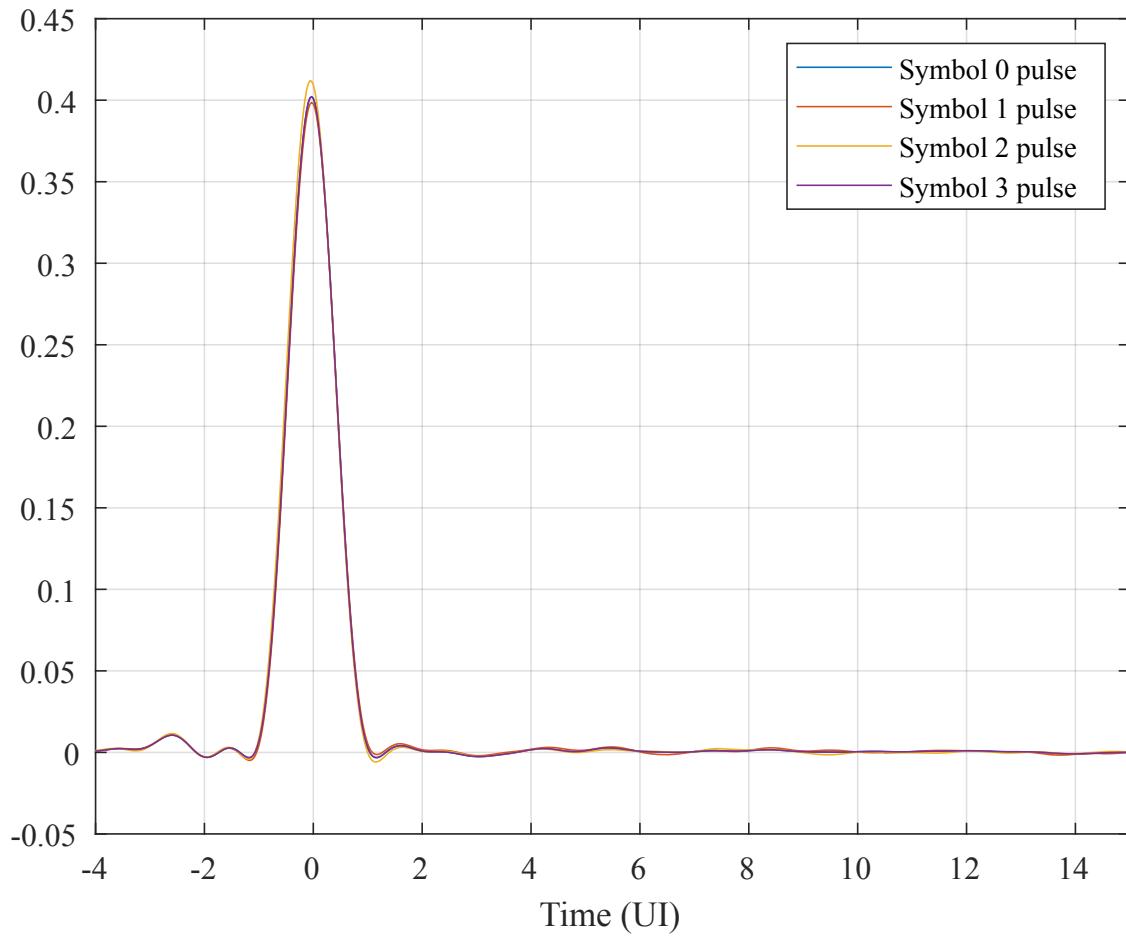


Figure 8-13 Example of Normalized Four Symbol Linear Pulse Responses §

The normalized pulses for all symbols are like each other as shown in § Figure 8-13 , make it easier to compare the pulses in the same scale. The normalized pulses can be un-normalized:

$$P_{sym0_un_normalized} = -P_{sym0_3}$$

$$P_{sym1_un_normalized} = -\frac{1}{3}P_{sym1}$$

$$P_{sym2_un_normalized} = \frac{1}{3}P_{sym2}$$

$$P_{sym3_un_normalized} = P_{sym0_3}$$

Equation 8-21 $P_{sym_un_normalized}$ §

The un-normalized pulses are shown in § [Figure 8-14](#).

Base 6.4 vs Base 6.3

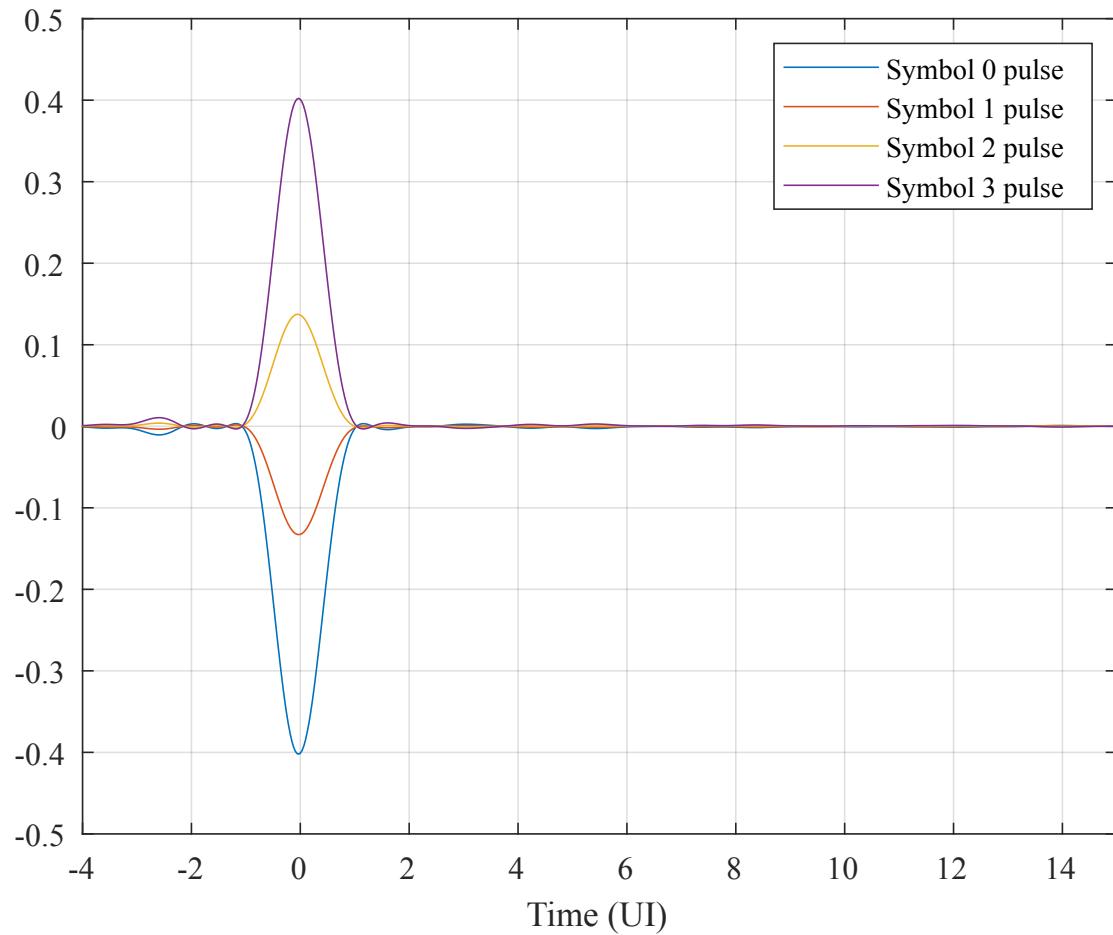


Figure 8-14 Example of Un-normalized Four Symbol Linear Pulse Responses §

The 4 pulses can be used to perform the R_{LM-TX} measurement § [Equation 8-17](#), by defining values V_L as the extrema of the un-normalized pulse fit vectors.

The multiple pulse linear fitting process is summarized as following:

1. Acquire the waveform from the transmitter.
2. Perform software clock recovery if needed, or hardware-based clock recovery.
3. Calculate the averaged data pattern waveform.
4. Get the data pattern sequence through pattern detection or obtained the known pattern sequence. Align the pattern sequence with the pattern waveform.
5. Construct the matrix X_{N_p} and Y based on the data pattern sequencer and the pattern waveform.
6. Construct the switch matrices for multiple pulse fitting.
7. Construct the matrices for the fitting solution based on the X_{N_p} and switch matrices.
8. Calculate the matrix P that contains pulses information.
9. Read the pulse samples $P_{pulse\ i}(k)$ from the matrix P .
10. Compute the un-normalized pulses using § Equation 8-21 .
11. Find the peaks of the un-normalized pulses which are used for V_0 , V_1 , V_2 , and V_3 .
12. Compute R_{LM-TX} using § Equation 8-17 .

8.3.4 Transmitter Margining §

Transmitters shall implement a margining procedure that allows the Tx launch voltage to be adjusted. Margining is enabled by programming a register set. Due to the larger range of Transmitter equalization, 8.0, 16.0, 32.0, and 64.0 GT/s Tx margining is subject to additional constraints: Tx margining at these speeds shall not require any coefficient or preset resolution finer than can be generated with 1/24 coefficient resolution defined for normal operation, and shall not require more Tx accuracy or capability than is required to support normal operation. It is acceptable that V_b fall below the limit set by $V_{TX-EIEOS-FS}$ or $V_{TX-EIEOS-RS}$, although proper end to-end operation is no longer guaranteed. Transmitter equalization accuracy requirements do not need to be met during margining. A Transmitter is not required to change the FS/LF values it sends in TS1 Ordered Sets during margining from the values used in normal operation.

There are 8 encoded values for transmit margin from 000b to 111b. Encoding 000b represents the normal operating range. For all supported data rates and Tx signaling mode (full swing or reduced swing), encoding 001b must produce a $V_{TX-DIFF-PP}$ compliant with the specification limits. At least three additional encodings with monotonically decreasing values for $V_{TX-DIFF-PP}$ must be supported for each data rate and Tx swing mode. For full swing signaling there must be at least one encoding with index 100b or higher that produces a $V_{TX-DIFF-PP}$ between 200 and 400 mV. For reduced swing signaling there must be at least one encoding with value 100b or higher that produces a $V_{TX-DIFF-PP}$ between 100 and 200 mV.

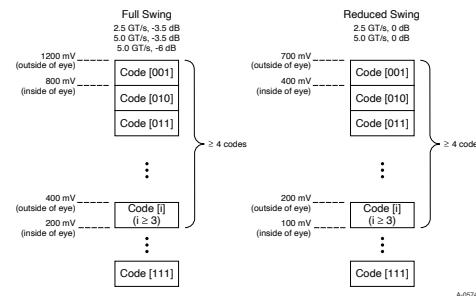


Figure 8-15 2.5 and 5.0 GT/s Transmitter Margining Voltage Levels and Codes §

8.3.5 Tx Jitter Parameters §

Jitter limits are defined identically for all data rates, although their respective values will vary with data rate. Jitter is measured at the zero-crossing point at full speed using the Compliance Pattern for 2.5, 5.0, 8.0, and 16.0 GT/s. When measuring jitter, the preset yielding the lowest jitter value should be selected. At 32.0 GT/s, the Tx under test must transmit Jitter Measurement Pattern (see § Section 4.2.13) with no Tx equalization. At 64.0 GT/s, the Tx under test must transmit Jitter Measurement Pattern (see § Section 4.2.16) with no Tx equalization for measuring the uncorrelated total jitter and deterministic jitter for all twelve transitions between the four PAM4 voltage levels. At 64.0 GT/s, the Tx under test must transmit High Swing Toggle Pattern (see § Section 4.2.17) with no Tx equalization for measuring the uncorrelated total pulse width jitter and deterministic pulse width jitter for the transitions between voltage level 0 and voltage level 3. When measuring a particular Tx Lane, it is necessary to ensure that all other PCI Express Lanes are transmitting Compliance Pattern in order to capture Tx die and package crosstalk effects. When measuring Tx jitter, it is required for the DUT to drive as many of its outputs as would occur during normal operation in a system environment. The minimum oscilloscope bandwidth for Tx jitter measurements at 32.0 and 64.0 GT/s is 50 GHz. The jitter measurements at 64.0 GT/s should be made with a 4th order Bessel-Thomson filter with a roll-off from DC value by 3 dB at 33 GHz to minimize the impact of scope high-frequency noise.

8.3.5.1 Post Processing Steps to Extract Jitter §

Measured Tx jitter is referenced to the Tx pin, and depending on what type of jitter is being measured and what reference clock architecture is being tested, is subsequently referenced to a recovered data clock, an embedded reference clock captured simultaneously with the data, or to a data edge. Data captured at TP1 requires post processing in order to remove the effects of the breakout channel and to regenerate a data clock (when an embedded reference clock is not captured simultaneously with the data).

8.3.5.2 Applying CTLE or De-embedding §

Direct probing at a Transmitter's pins is not generally feasible, so data is instead measured at TP1 of the breakout channel. By means of the replica channel it is possible to determine the loss vs. frequency characteristics of the breakout channel and de-embed this channel, resulting in measurements that are effectively referenced to the DUT's pins. Note that since de-embedding amplifies HF noise there is a practical frequency cutoff limit to de-embedding. As de-embedding amplifies HF channel and measurement noise, an HF cutoff limit must be applied to de-embedding, depending on data rate as shown in § Table 8-4.

Table 8-4 Recommended De-embedding Cutoff Frequency §

| Data Rate | HF Cutoff limit for de-embedding |
|-----------|----------------------------------|
| 8.0 GT/s | 8 GHz - 12 GHz |
| 16.0 GT/s | 20 GHz |

Jitter is decomposed into data dependent and uncorrelated terms. This separation process effectively separates the jitter caused by package effects from that caused by silicon effects such as PLL jitter and power supply noise that cannot be mitigated by equalization. As a result, the uncorrelated jitter terms define jitter as it would appear at the die pad.

As an alternative to de-embedding at 16.0 GT/s the -12 dB CTLE in the reference equalizer can be applied to the data measured at TP1 for measuring all uncorrelated jitter parameters (not DDJ).

It is recommended that s-parameters for de-embedding are measured to at least 3 times the Nyquist frequency with a frequency step size of 10 MHz.

If both de-embedding and CTLE approaches are used and given different answers only the lower values for the uncorrelated jitter parameters are used.

For 32.0 GT/s, Jitter Measurement Pattern (see § [Section 4.2.13](#)) with no Tx equalization is used to minimize the breakout channel ISI impact and avoid pessimism in jitter measurement from de-embedding and associated high frequency scope noise amplification. For further reduction of channel loss impact on jitter at 32.0 GT/s, any CTLE curve in the reference equalizer or no CTLE curve can be applied to the data measured at TP1 for measuring all uncorrelated jitter parameters (not DDJ). The CTLE or no CTLE curve that gives the lowest result for $T_{TX-UPW-TJ}$ is used.

For 64.0 GT/s, the Jitter Measurement Pattern (see § [Section 4.2.16](#)) with no Tx equalization is used for measuring the uncorrelated total jitter and the uncorrelated deterministic jitter for all twelve transitions between the four PAM4 voltage levels. The 64.0 GT/s Jitter Measurement Pattern is 52-UI long and all 12 PAM4 transitions repeat four times within the pattern resulting in 48 edge transitions. Jitter must be measured on each of the 48 edges individually and then averaged.

The Q-scale associated with the 64.0 GT/s Jitter Measurement Pattern for BER of 10^{-6} is 4.8759. For mitigating channel ISI impact on jitter at 64.0 GT/s, any CTLE curve in the reference equalizer or no CTLE curve can be applied to the data measured at TP1 for measuring all uncorrelated jitter parameters (not DDJ). The CTLE or no CTLE curve that gives the lowest result for the average T_{TX-RJ} of the 48 edges is used. The uncorrelated jitter parameters for all 48 transitions must be measured and corrected for the scope noise impact. The average uncorrelated total jitter and the average uncorrelated deterministic jitter are obtained by averaging the jitter parameters of all 48 edge transitions.

For 64.0 GT/s, the High Swing Toggle Pattern (see § [Section 4.2.17](#)) with no Tx equalization is used for measuring the uncorrelated total pulse width jitter and deterministic pulse width jitter for transitions between voltage level 0 and voltage level 3. The Q-scale associated with the 64.0 GT/s High Swing Toggle Pattern for BER of 10^{-6} is 4.8916. The High Swing Toggle Pattern minimizes the breakout channel ISI impact and avoids pessimism in jitter measurement from de-embedding and associated high frequency scope noise amplification. For further reduction of channel loss impact on pulse width jitter at 64.0 GT/s, any CTLE curve in the reference equalizer or no CTLE curve can be applied to the data measured at TP1 for measuring all uncorrelated jitter parameters (not DDJ). The CTLE or no CTLE curve that gives the lowest result for $T_{TX-UPW-TJ}$ is used. The uncorrelated pulse width jitter parameters must be corrected for the scope noise impact.

8.3.5.3 Independent Refclk Measurement and Post Processing §

A Transmitter may operate in the Independent Refclk (IR) mode, in which case the Transmitter may not provide a Refclk output. In this case a single-port jitter measurement is required. The post processing algorithm must employ the appropriate model CDR for the reference clock architecture being tested.

8.3.5.4 Embedded and Non-Embedded Refclk Measurement and Post Processing §

When the transmitting PCIe device is driven from an external source to its Refclk pin it permits the Tx under test to be driven with a clean Refclk as shown in § [Figure 8-1](#).

The specification now explicitly supports the complete matrix of Refclk options, including where the Refclk is embedded, where the reference clock is external, where the reference clock is available at the DUT's pins, and where the reference clock is not available at the DUT's pins. § [Table 8-5](#) lists the post processing requirements for each of the four possible combinations. Embedded Refclk with Refclk available at the DUT's pin represents a special case where any jitter common to both the Refclk and the data must be removed via a two-port measurement.

If the DUT supports multiple Refclk modes, as described in § [Table 8-5](#) then the Tx needs to be tested in each of the Refclk modes it supports.

Table 8-5 Tx Measurement and Post Processing For Different Refclks §

| | Embedded Refclk | Non-Embedded Refclk |
|---|---|---|
| Refclk available at DUT pin and not testing SRIS mode | 2-port measurement, CC CDR, PLL ¹ and 10 ns transport delay ² | 1-port measurement, CC CDR, clean external Refclk |
| Refclk not available at DUT pin or testing SRIS mode | 1-port measurement, SRIS CDR | 1-port measurement, CC CDR, clean external Refclk |

Notes:

1. PLL characteristics are defined in Refclk section for each data rate.
2. Refer to § Section 8.6.6 for a discussion of the transport delay

8.3.5.5 Behavioral CDR Characteristics §

A behavioral CDR filter is applied to reject low frequency jitter that would normally be tracked by the CDR in a Receiver. As such, the behavioral CDR represents a bounding function for actual CDR implementations. Roll-off characteristics of the behavioral CDR are dependent on whether the corresponding DUT supports an embedded vs. non-embedded Refclk, is operating in CC or IR mode, and whether the Refclk pin is available for probing (see § Table 8-5). In all cases the behavioral CDR represents a high pass filter function where the corner frequency depends on the Tx data rate. § Figure 8-16 shows the CC first-order CDR transfer functions for an f_{3dB} of 1.5 MHz, 5.0 MHz, and 10 MHz that corresponds to 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s, respectively. The 10 MHz behavioral CDR is also used for CC Transmitter and CC Reference Clock testing for 16.0 GT/s and, optionally, for Receiver Stressed Eye calibration when 32.0 GT/s is not supported.

Base 6.4 vs Base 6.3

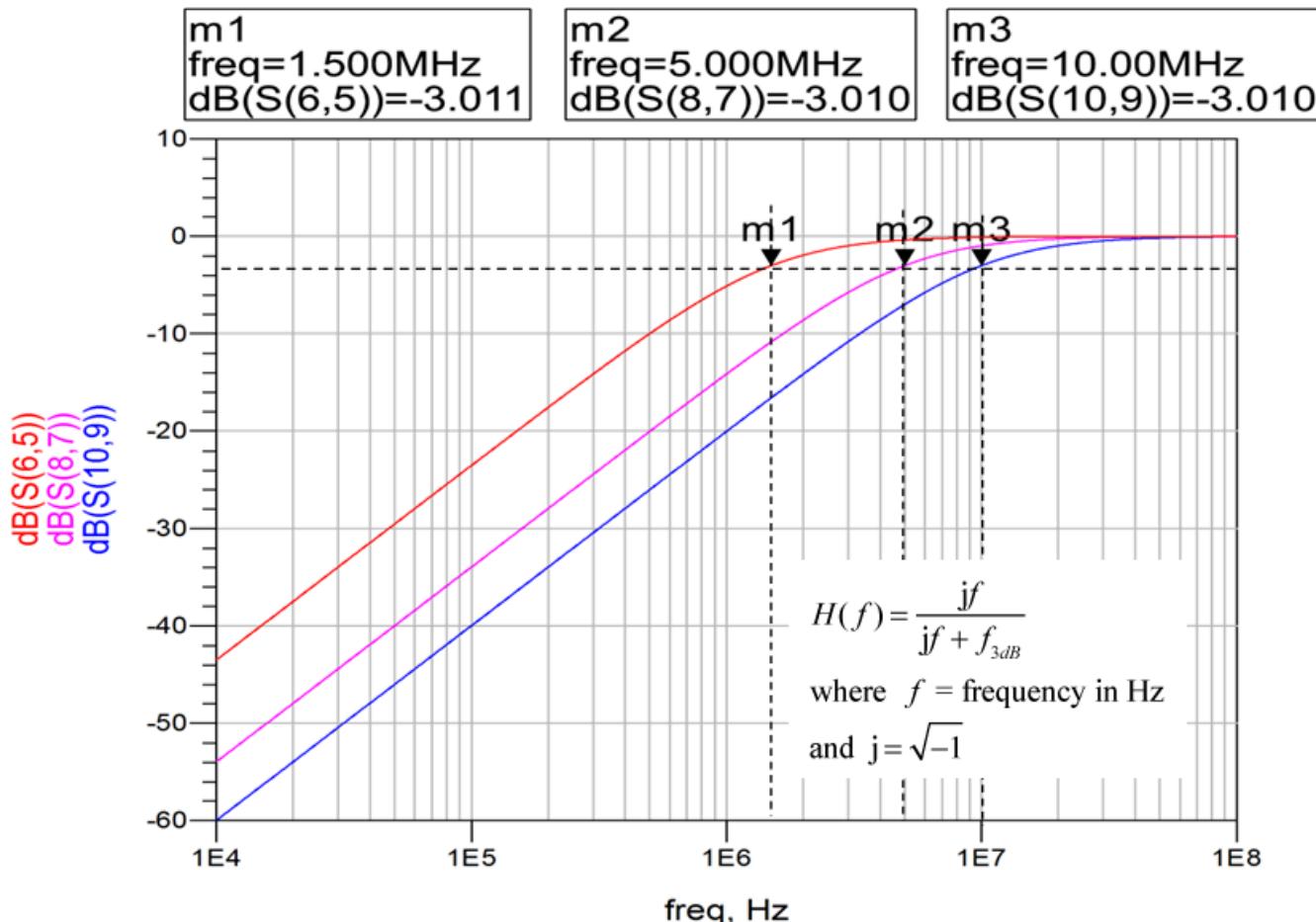


Figure 8-16 First Order CC Behavioral CDR Transfer Functions §

§ Figure 8-17 illustrates second order CDR transfer functions corresponding to 2.5 GT/s and 5.0 GT/s. These functions are defined by a ζ of 0.707 and an f_{3dB} of 1.5 MHz and 5.0 MHz, respectively. Behavioral transfer functions for 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s approximate the piecewise linear sinusoidal jitter (S_j) masks shown in § Section 8.4.2.2.1 . SRIS capable Transmitters must be evaluated using these behavioral transfer functions.

Note: The common clock (CC) and independent reference clock (IR) architectures are not interoperable - although it is possible to design a single Receiver that meets both sets of electrical requirements.

Base 6.4 vs Base 6.3

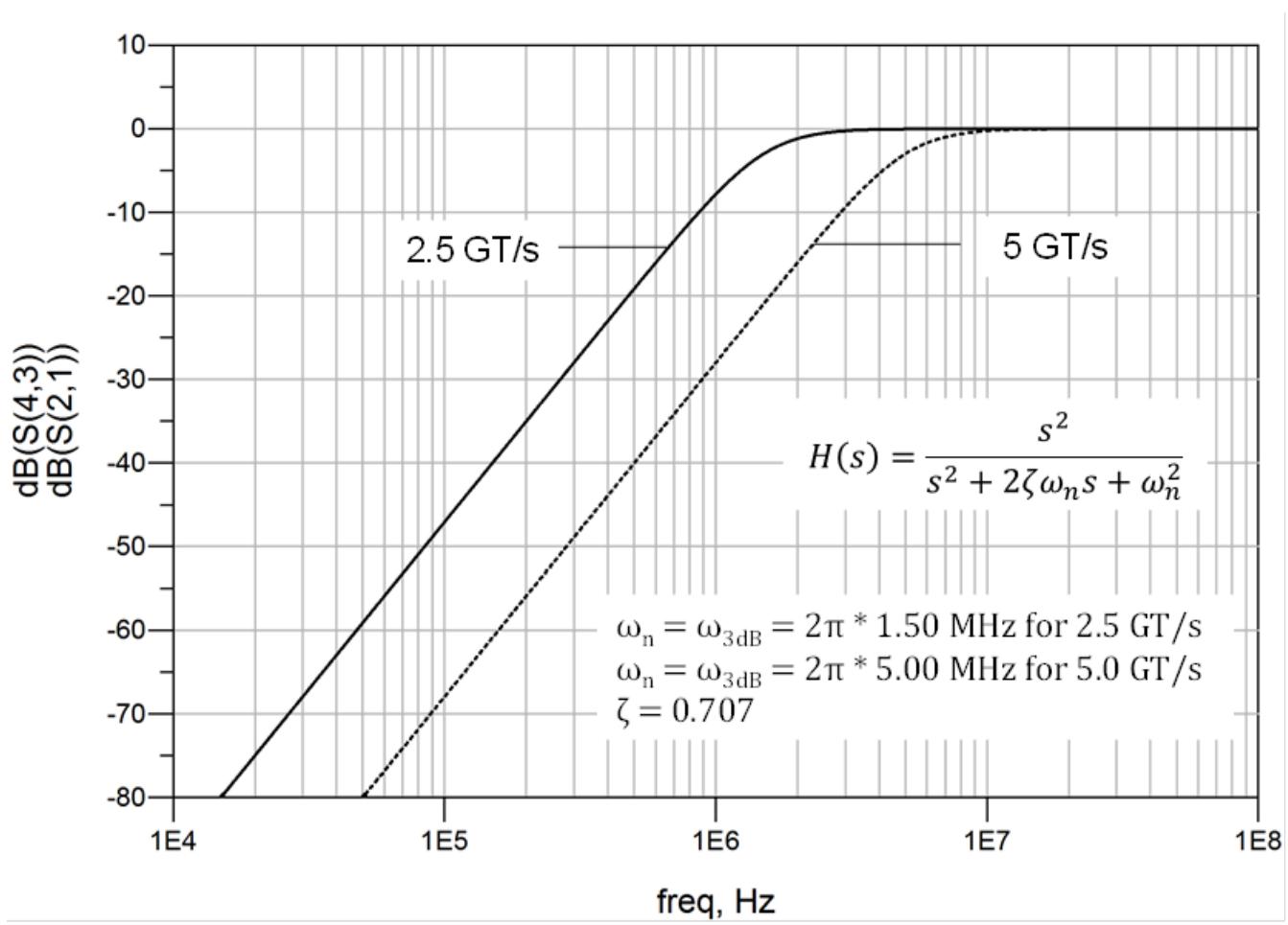


Figure 8-17 2nd Order Behavioral SRIS CDR Transfer Functions for 2.5 GT/s and 5.0 GT/s §

Base 6.4 vs Base 6.3

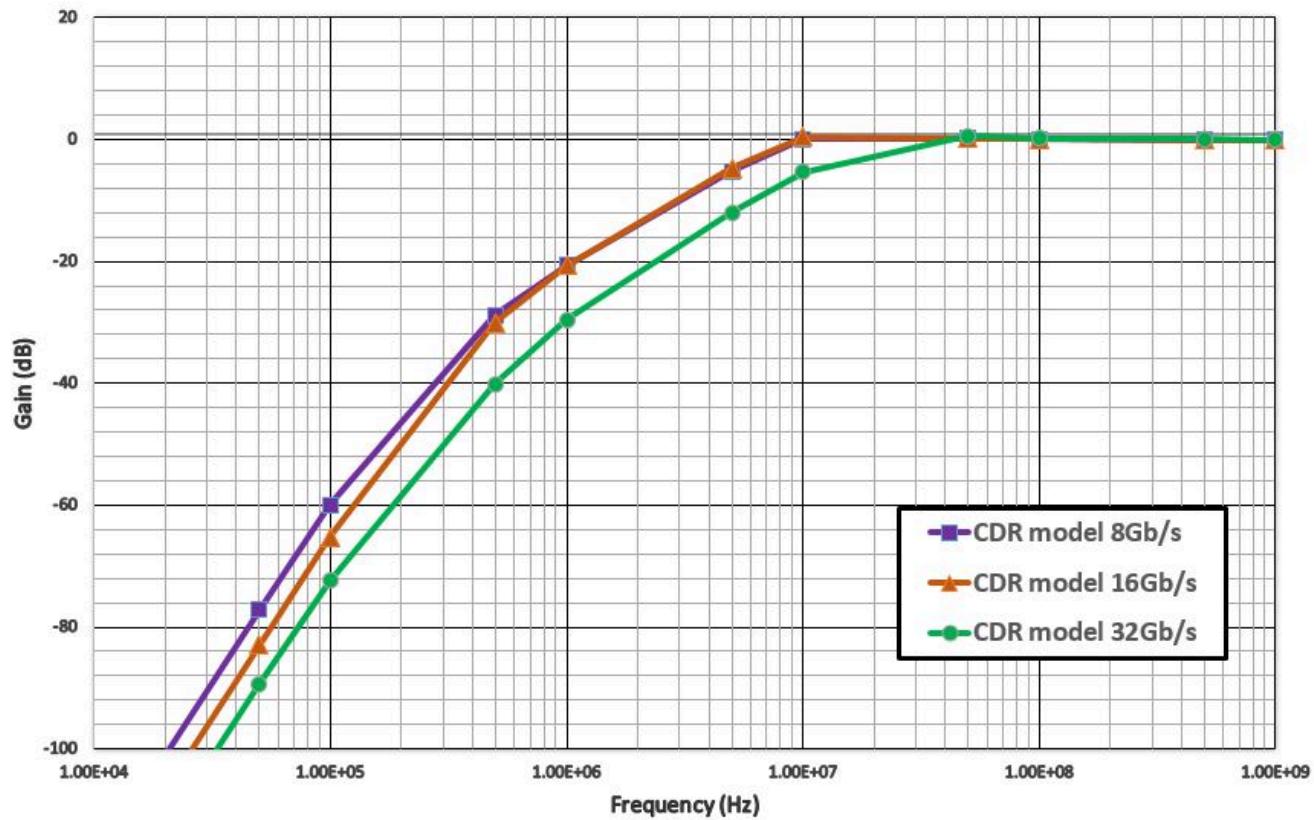


Figure 8-18 Behavioral SRIS CDR Function for 8.0 GT/s, and SRIS and CC CDR for 16.0 and 32.0 GT/s §

$$H(s) = \frac{s^2}{s^2 + sA + B} \times \frac{s^2 + 2\zeta_2\omega_0s + \omega_0^2}{s^2 + 2\zeta_1\omega_0s + \omega_0^2} \times \frac{s}{s + \omega_1}$$

$$\zeta_1 = \frac{1}{\sqrt{2}}$$

$$\zeta_2 = 1$$

$$\omega_0 = 10^7 \times 2\pi$$

$$\omega_1 = 4 \times 10^5 \times 2\pi$$

Equation 8-22 Behavioral SRIS CDR at 8.0 GT/s and SRIS and CC Behavioral CDR at 16.0 GT/s §

$$A = 10^7 \times 2\pi$$

$$B = 2.2 \times 10^{12} \times (2\pi)^2$$

Equation 8-23 SRIS Behavioral CDR Parameters at 8.0 GT/s §

$$A = 9.5 \times 10^6 \times 2\pi$$

$$B = 4.36 \times 10^{12} \times (2\pi)^2$$

Equation 8-24 SRIS and CC Behavioral CDR Parameters at 16.0 GT/s §

Base 6.4 vs Base 6.3

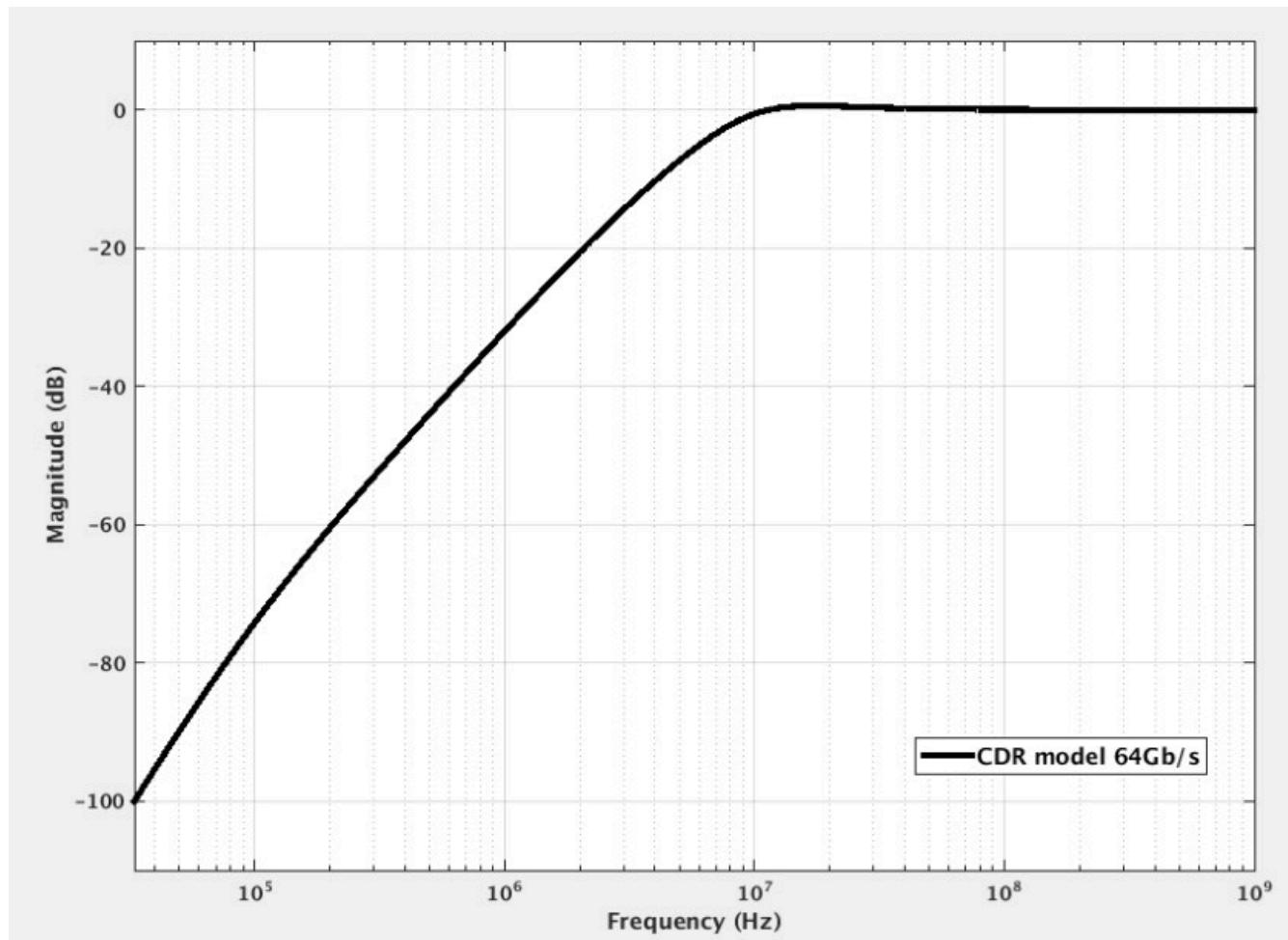


Figure 8-19 Behavioral SRIS and CC CDR for 64.0 GT/s §

$$H(s) = \frac{s^2}{(s + \omega_0)(s + \omega_1)} \times \frac{s^2 + 2\zeta_2\omega_0 s + \omega_0^2}{s^2 + 2\zeta_1\omega_0 s + \omega_0^2} \times \frac{s}{s + \omega_{LF}}$$

$$\zeta_1 = \frac{1}{\sqrt{2}}$$

$$\zeta_2 = 1$$

32.0 GT/s

64.0 GT/s

$$\omega_0 = 20 \times 10^6 \times 2\pi$$

$$\omega_0 = 10 \times 10^6 \times 2\pi$$

$$\omega_1 = 1.1 \times 10^6 \times 2\pi$$

$$\omega_1 = 3.88 \times 10^6 \times 2\pi$$

$$\omega_{LF} = 160 \times 10^3 \times 2\pi$$

$$\omega_{LF} = 87 \times 10^3 \times 2\pi$$

Equation 8-25 SRIS and CC Behavioral CDR Parameters at 32.0 and 64.0 GT/s

8.3.5.6 Data Dependent and Uncorrelated Jitter §

Measured at TP1 and de-embedded back to the pin, a Transmitter's jitter contains both data dependent and uncorrelated components. The data dependent components occur principally due to package loss and reflection. Uncorrelated jitter sources include PLL jitter, power supply noise, and crosstalk. The specification separates jitter into uncorrelated and data dependent bins, because such a separation matches well with the Tx and Rx equalization capabilities. Uncorrelated jitter is not mitigated by Tx or Rx equalization and represents timing margin that cannot be recovered via equalization. It is important that margin recoverable by means of equalization (data dependent) is not budgeted as non-recoverable jitter.

Once data dependent jitter has been removed from the Tx measurement it becomes possible to resolve the remaining jitter into T_j and deterministic jitter (Dual Dirac Model) (DJDD) components. High frequency jitter (which is subject to jitter amplification in the channel) is accounted for by separate $T_{TX-UPW-DJDD}$ and $T_{TX-UPW-T_j}$ parameters.

8.3.5.7 Data Dependent Jitter §

While DDJ is not explicitly defined as a parameter in the specification, it is necessary to separate DDJ in order to eliminate package loss effects and reference the jitter parameters of interest to the Tx die pad. Separation of jitter into data dependent and uncorrelated components may be achieved by averaging techniques; for example, by having the Tx repeatedly drive the compliance test pattern which is a repeating pattern.

§ Figure 8-20 illustrates the relation between Tx data, recovered clock, and the data's PDF. Data dependent jitter is defined as the time delta between the PDF's mean for each zero-crossing point and the corresponding recovered clock edge. A sufficient number of repeated patterns must be accumulated to yield stable mean values and PDF profiles for each transition. These PDFs are then utilized to extract uncorrelated jitter parameters.

Base 6.4 vs Base 6.3

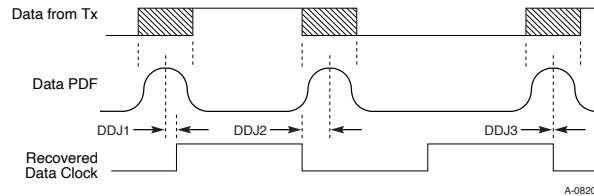


Figure 8-20 Relation Between Data Edge PDFs and Recovered Data Clock §

8.3.5.8 Uncorrelated Total Jitter and Deterministic Jitter (Dual Dirac Model) (T_{TX-UTJ} and $T_{TX-UDJDD}$) §

Uncorrelated Total Jitter (UTJ) and uncorrelated deterministic jitter (Dual Dirac model) (UDJDD) are referenced to a recovered data clock generated by means of a CDR tracking function. Uncorrelated jitter may be derived after removing the DDJ component from each PDF and combining the PDFs for all edges in the pattern. By appropriately converting the PDF to a Q-scale it is possible to obtain the graphical relation shown in § Figure 8-21 , from which T_{TX-UTJ} and $T_{TX-UDJDD}$ may be derived. In § Figure 8-21 note that the two PDF curves are identical but that the fitted slopes, defined by $1/RJ_{LH}$ and $1/RJ_{RH}$, may differ.

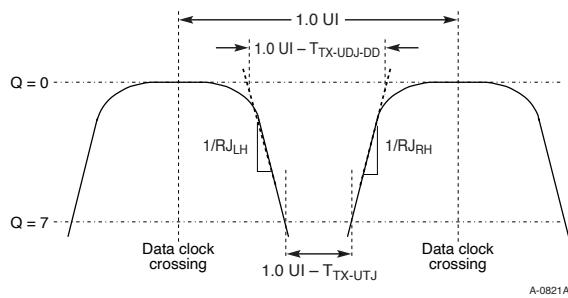


Figure 8-21 Derivation of T_{TX-UTJ} and $T_{TX-UDJDD}$ §

8.3.5.9 Random Jitter (T_{TX-RJ}) (informative) §

Random jitter is uncorrelated with respect to data dependent jitter. T_{TX-RJ} may be obtained by subtracting $T_{TX-UDJDD}$ from T_{TX-UTJ} and is included in the specification as an informative parameter only. It is typically used as a benchmark to characterize PLL performance.

8.3.5.10 Uncorrelated Total and Deterministic PWJ ($T_{TX-UPW-TJ}$ and $T_{TX-UPW-DJDD}$) §

Pulse width jitter is defined as an edge to edge phenomenon on consecutive edges nominally 1.0 UI apart. § Figure 8-22 illustrates how PWJ is defined, showing that it is typically present on both data edges of consecutive UI . To accurately quantify PWJ it is first necessary to remove the ISI contributions to PWJ. The shaded areas on either side of the unjittered edges represent the maximum amount of jitter about that edge. Note the jitter for one edge is assumed to be independent from the other.

Base 6.4 vs Base 6.3

An equivalent description of PWJ may be obtained by referencing to a fixed leading edge and having jitter contributions from both edges appear at the trailing edge. This approach yields a single PDF as shown below. Each 1 UI wide pulse in the pattern will have a different median for this PDF which is caused by ISI and F/2 jitter. The average of the medians for 1 UI wide pulses at odd and even UI numbers within the pattern are calculated, and the odd and even PDF's are normalized to the appropriate average of medians and summed to form an odd UI PDF and an even UI PDF. The final PDF is calculated from the sum of the summed odd and even UI PDFs. The key idea here is that the final PDF for uncorrelated PWJ should include F/2 or odd/even UI jitter

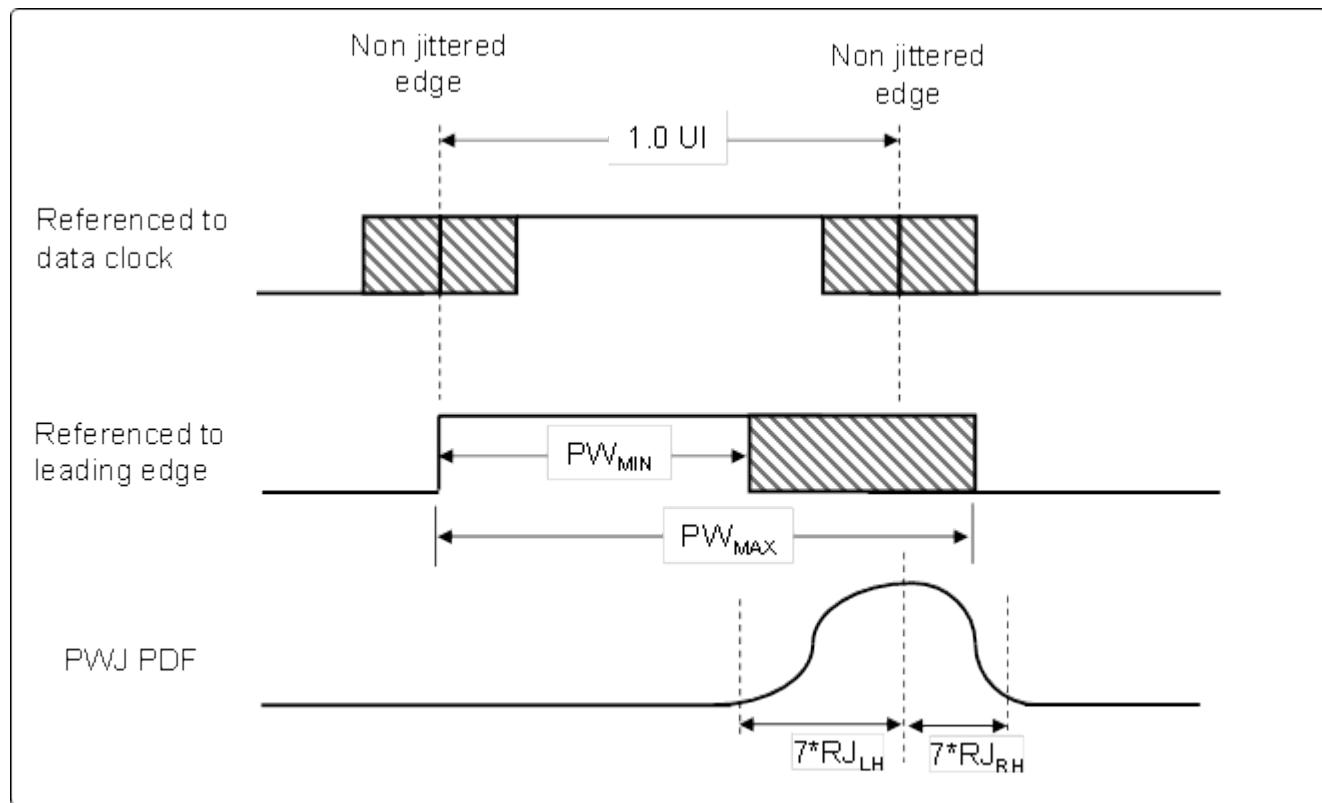


Figure 8-22 PWJ Relative to Consecutive Edges 1 UI Apart §

The PDF of jitter around each non-jittered edge may be converted into the Q-scale (see § Figure 8-23) from which $T_{TX-UPW-TJ}$ and $T_{TX-UPW-DJDD}$ may be derived in a manner analogous to T_{TX-UTJ} and $T_{TX-UDJDD}$. Note that the PDF may not be symmetric, and the tail of interest is RJ_{LH} , since it represents pulse compression.

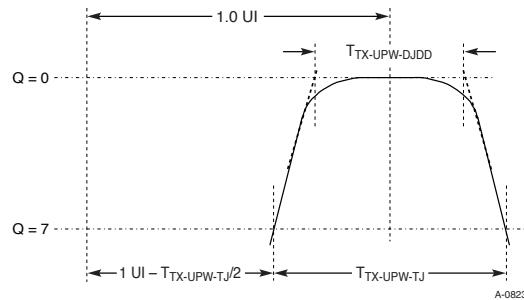


Figure 8-23 Definition of $T_{TX-UPW-DJDD}$ and $T_{TX-UPW-TJ}$ Data Rate Dependent Transmitter Parameters §

8.3.6 Data Rate Dependent Parameters §

Note: the jitter margins for 2.5 GT/s and 5.0 GT/s were previously defined at the device's pins. For 2.5 GT/s the jitter was defined via a single parameter that lumped DDJ, UDJDD, UTJ, PWJ-DDJ and PWJ-TJ into a single quantity. Consequently, it is necessary to first remove the DDj jitter component. Since there was no previous UDj-UTj separation T_{TX-UTJ} and $T_{TX-UDJDD}$ are set equal to each other. Similarly, there was no UTj-PWj separation, so it is necessary to assume that the entirety of the uncorrelated jitter is PWJ that occurs oppositely on consecutive edges of a 1 UI wide pulse.

For 5.0 GT/s a similar removal of DDj must be performed to obtain UTj. However, [PCIe-3.0] for 5.0 GT/s did specify Rj, so a distinct value for $T_{TX-UDJDD}$ can be obtained. Similarly, [PCIe-3.0] for 5.0 GT/s defined a minimum pulse width, assumed to be 100% Dj, from which $T_{TX-UPW-TJ}$ and $T_{TX-UPW-DJDD}$ may be derived. For 64.0 GT/s PAM4 signaling, the voltage parameters such as differential peak-to-peak Tx voltage swing correspond to the swing between PAM4 voltage level 0 and voltage level 3.

Table 8-6 Data Rate Dependent Transmitter Parameters §

| Symbol | Parameter description | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s | Units | Notes |
|----------------------|--|---|---|---|---|---|---|-------|--|
| $UI(Tx)$ | Unit Interval | (min) 399.88 (max) 400.12 (300 PPM) | (min) 199.94 (max) 200.06 (300 PPM) | (min) 124.9625 (max) 125.0375 (300 PPM) | (min) 62.48125 (max) 62.51875 (300 PPM) | (min) 31.246875 (max) 31.253125 (100 PPM) | (min) 31.246875 (max) 31.253125 (100 PPM) | ps | Does not include SSC variations |
| $BW_{TX-PKG-PLL1}$ | Tx PLL bandwidth corresponding to $PKG_{TX-PLL1}$ | (min) 1.5 (max) 22.0 | (min) 8.0 (max) 16.0 | (min) 0.5 (max) 4.0 | (min) 0.5 (max) 4.0 | (min) 0.5 (max) 1.8 | (min) 0.5 (max) 1.0 | MHz | Second order PLL jitter transfer bounding function. Notes 1, 2, 9. |
| $BW_{TX-PKG-PLL2}$ | Tx PLL bandwidth corresponding to $PKG_{TX-PLL2}$ | N/A | (min) 5.0 (max) 16.0 | (min) 0.5 (max) 5.0 | (min) 0.5 (max) 5.0 | N/A | N/A | MHz | 2.5 and 32.0 GT/s specify only one combination of PLL BW and jitter. Notes 1, 2, 9. |
| $PKG_{TX-PLL1}$ | Tx PLL peaking corresponding to $BW_{TX-PKG-PLL1}$ | (max) 3.0 | (max) 3.0 | (max) 2.0 | (max) 2.0 | (max) 2.0 | (max) 2.0 | dB | Second order PLL jitter transfer bounding function. Notes 1, 2. |
| $PKG_{TX-PLL2}$ | Tx PLL peaking corresponding to $BW_{TX-PKG-PLL2}$ | N/A | (max) 1.0 | (max) 1.0 | (max) 1.0 | N/A | N/A | dB | 2.5 and 32.0 GT/s specify only one combination of PLL BW and jitter. Notes 1, 2. |
| $V_{TX-DIFF-PP}$ | Differential peak-peak Tx voltage swing for full swing operation | (min) 800 (max) 1000 | (min) 800 (max) 1000 | (min) 800 (max) 1000 | (min) 800 (max) 1000 | (min) 800 (max) 1000 | (min) 800 (max) 1000 | mVPP | As measured with compliance test load. Defined as $2 \times V_{TXD+} - V_{TXD-} $. Note 3. |
| $V_{TX-DIFF-PP-LOW}$ | Differential peak-peak Tx | (min) 400 | (min) 400 | (min) 400 | (min) 400 | (min) 400 | (min) 400 | mVPP | As measured with compliance test |

Base 6.4 vs Base 6.3

| Symbol | Parameter description | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s | Units | Notes |
|------------------------------|--|---------------|---------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|-------|--|
| | voltage swing for low swing operation | (max) 1000 | (max) 1000 | (max) 1000 | (max) 1000 | (max) 1000 | (max) 1000 | | load. Defined as $2 \times V_{TXD+} - V_{TXD-} $ Note 3. |
| $V_{TX-EIEOS-FS}$ | Minimum voltage swing during EIEOS for full swing signaling | N/A | N/A | 250 (min) | 250 (min) | 250 (min) | 250 (min) | mVPP | Note 4 |
| $V_{TX-EIEOS-RS}$ | Minimum voltage swing during EIEOS for reduced swing signaling | N/A | N/A | 232 (min) | 232 (min) | 232 (min) | 232 (min) | mVPP | Note 4 |
| $ps21_{TX-ROOT-DEVICE}$ | Pseudo package loss of a device containing root port | N/A | N/A | (max) 3.0 | (max) 5.0 | (max) 8.5 | (max) 7.5 | dB | Note 5. |
| $ps21_{TX-NON-ROOT-DEVICE}$ | Pseudo package loss for all devices not containing root ports | N/A | N/A | (max) 3.0 | (max) 3.0 | (max) 3.7 | (max) 3.7 | dB | Note 5. |
| $ILfit_{TX-ROOT-DEVICE}$ | Fitted insertion loss at Nyquist | N/A | N/A | N/A | N/A | (max) 9.0 | (max) 8.0 | dB | Note 8 |
| $ILfit_{TX-NON-ROOT-DEVICE}$ | Fitted insertion loss at Nyquist | N/A | N/A | N/A | N/A | (max) 4.0 | (max) 4.0 | dB | Note 8 |
| $V_{TX-BOOST-FS}$ | Maximum nominal Tx boost ratio for full swing | N/A | N/A | 8.0 | 8.0 (min) | 8.0 (min) | 8.0 (min) | dB | Nominal boost beyond 8.0 dB is limited to guarantee that $ps21_{TX}$ limits are satisfied. |
| $V_{TX-BOOST-RS}$ | Maximum nominal Tx boost ratio for reduced swing | N/A | N/A | 2.5 | ~2.5 (min) | ~2.5 (min) | ~2.5 (min) | dB | Assumes ± 1.0 dB tolerance from diagonal elements in § Figure 8-9. |
| $EQ_{TX-CO-EFF-RES}$ | Tx coefficient resolution | N/A | N/A | 1/(min) 63 1/(max) 24 | 1/(min) 63 1/(max) 24 | 1/(min) 63 1/(max) 24 | 1/(min) 63 1/(max) 24 | N/A | |

Base 6.4 vs Base 6.3

| Symbol | Parameter description | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s | Units | Notes |
|--------------------------|--|------------------------------|------------------------------|----------------|----------------|----------------|-----------------------------|---------------------|--|
| $V_{TX-DE-RA-TIO-3.5dB}$ | Tx de-emphasis ratio for 2.5 and 5.0 GT/s | (min) 2.5 (max) 4.5 | (min) 2.5 (max) 4.5 | N/A | N/A | N/A | N/A | dB | |
| $V_{TX-DE-RA-TIO-6dB}$ | Tx de-emphasis ratio for 5.0 GT/s | N/A | (min) 4.5 (max) 7.5 | N/A | N/A | N/A | N/A | dB | |
| T_{TX-UTJ} | Tx uncorrelated total jitter | (max) 100 | (max) 50 | (max) 27.55 | (max) 11.8 | (max) 6.25 | (max) 4.00 at 10^{-6} | ps PP at 10^{-12} | See ¶ and ¶ § Section 8.3.5.1 ¶ and ¶ § Section 8.3.5.8 for details. |
| $T_{TX-UTJ-SRIS}$ | Tx uncorrelated total jitter when testing for the IR clock mode with SSC | (max) 100 | (max) 66.51 | (max) 33.83 | (max) 15.85 | (max) 7.15 | (max) 4.389 at 10^{-6} | ps PP at 10^{-12} | See ¶ and ¶ § Section 8.3.5.1 ¶ and ¶ § Section 8.3.5.8 for details. |
| $T_{TX-UDJDD}$ | Tx uncorrelated Dj for non-embedded Refclk | (max) 100 | (max) 30 | (max) 12 | (max) 6.25 | (max) 3.125 | (max) 1.563 | ps PP | See ¶ and ¶ § Section 8.3.5.1 ¶ and ¶ § Section 8.3.5.8 for details. |
| $T_{TX-UPW-TJ}$ | Total uncorrelated pulse width jitter | N/A | (max) 40 | (max) 24 | (max) 12.5 | (max) 6.25 | (max) 4.00 at 10^{-6} | ps PP at 10^{-12} | See ¶ and ¶ § Section 8.3.5.1 ¶ and ¶ § Section 8.3.5.8 for details. |
| T_{TX-RJ} | Tx Random jitter | N/A | 1.4 - 3.6 | 1.17 - 1.97 | 0.40 - 0.84 | 0.23 - 0.45 | 0.26 - 0.42 | ps RMS | Informative parameter only. Range of Rj possible with zero to maximum allowed $T_{TX-UDJDD}$. |
| $T_{TX-UPW-DJDD}$ | Deterministic DjDD uncorrelated pulse width jitter | N/A | (max) 40 | (max) 10 | (max) 5 | (max) 2.5 | (max) 1.25 | ps PP | See ¶ and ¶ § Section 8.3.5.1 ¶ and ¶ § Section 8.3.5.8 for details. |
| R_{LM-TX} | Level Separation Mismatch Ratio | N/A | N/A | N/A | N/A | N/A | (min) 0.95 | | See § Section 8.3.3.14 for details |
| $SNDR_{TX}$ | Signal-to-Noise-Distortion Ratio | NA | NA | NA | NA | NA | (min) 34 | dB | See § Section 8.3.3.13 for details |

Base 6.4 vs Base 6.3

| Symbol | Parameter description | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s | Units | Notes | |
|----------------------------|--|-------------------|--------------|--------------|---------------|---------------|---------------|-------------------|--|--------|
| $V_{TX-AC-CM-PP}$ | Tx AC peak-peak common mode voltage over 0.03-500 MHz range | (max) 150 | (max) 100 | (max) 50 | (max) 50 | (max) 50 | (max) 25 | mVPP | Tx ACCM noise measurement analysis is done without any de-embedding. | |
| $V_{TX-AC-CM-PP-filtered}$ | Tx AC peak-peak common mode voltage filtered with a simple low-pass filter (-3 dB roll-off at Nyquist frequency) | (max) 150 | (max) 150 | (max) 150 | (max) 150 | (max) 150 | (max) 75 | mVPP | Tx ACCM noise measurement analysis is done without any de-embedding. | |
| $L_{TX-SKEW}$ | Lane-to-Lane Output Skew | (max) 2.5 | (max) 2.0 | (max) 1.5 | (max) 1.25 | (max) 1.25 | (max) 1.25 | ns | Between any two Lanes within a single Transmitter. | |
| $RL_{TX-DIFF}$ | Tx package plus die differential return loss | See § Figure 8-26 | | | | | | See § Figure 8-26 | dB | Note 6 |
| RL_{TX-CM} | Tx package plus die common mode return loss | See § Figure 8-25 | | | | | | See § Figure 8-27 | dB | Note 6 |

Notes:

1. A single combination of PLL BW and peaking is specified for 2.5, 32.0, and 64.0 GT/s implementations. For other data rates, two combinations of PLL BW and peaking are specified to permit designers to make a tradeoff between the two parameters.
2. The Tx PLL Bandwidth must lie between the min and max ranges given in the above table. PLL peaking must lie below the value listed above. Note: the PLL B/W extends from zero up to the value(s) specified in the above table. The PLL BW is defined at the point where its transfer function crosses the -3dB point.
3. See § Section 8.3.3.6 and § Section 8.3.3.7 for measurement details.
4. $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ are measured at the device pin and include package loss. Voltage limits comprehend both full swing and reduced swing modes. A Transmitter must advertise a value for LF during TS1 at 8.0, 16.0, 32.0, and 64.0 GT/s that ensures that these parameters are met.
5. The numbers above consider measurement error. For some Tx package/driver combinations $ps21_{TX}$ may be greater than 0 dB. The channel compliance methodology at 2.5 and 5.0 GT/s assumes the 8.0 GT/s package model.
6. The DUT must be powered up and DC isolated, and its data+/data- outputs must be in the low-Z state at a static value.
7. The reference plane for all parameters at 2.5 and 5.0 GT/s is the package pins.
8. These are design parameter requirements - a specific test methodology for them is not defined.
9. For PCIe 6.0 devices that do not support 32.0 and 64.0 GT/s have the option to use 2 MHz as min of $BW_{TX-PKG-PLL1}$ and $BW_{TX-PKG-PLL2}$ for both 8.0 and 16.0 GT/s. The corresponding T_{TX-UTJ} max value is 31.25 ps at 8.0 GT/s and 12.5 ps at 16.0 GT/s. The range of T_{TX-RJ} is 1.4-2.2 ps at 8.0 GT/s and 0.45-0.89 ps at 16.0 GT/s. Such devices also have the option to use 1st-order, 10 MHz CDR filter for testing Tx, Reference clock, and CC Rx.

8.3.7 Tx and Rx Return Loss for 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s §

Return loss measurements for the Tx and Rx are essentially identical, so both are included in the Transmitter section. Return loss measurements are made at the end of the respective breakout channels and require that the breakout channel's contribution to RL be de-embedded, thereby associating the return loss with the Tx or Rx pin. Return loss measurements are made with a reference impedance of 50 ohms. § Figure 8-24 defines the pass/fail mask for differential return loss for 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s. Both differential and common mode are defined over a frequency range of 50 MHz to 16.0 GHz.

Base 6.4 vs Base 6.3

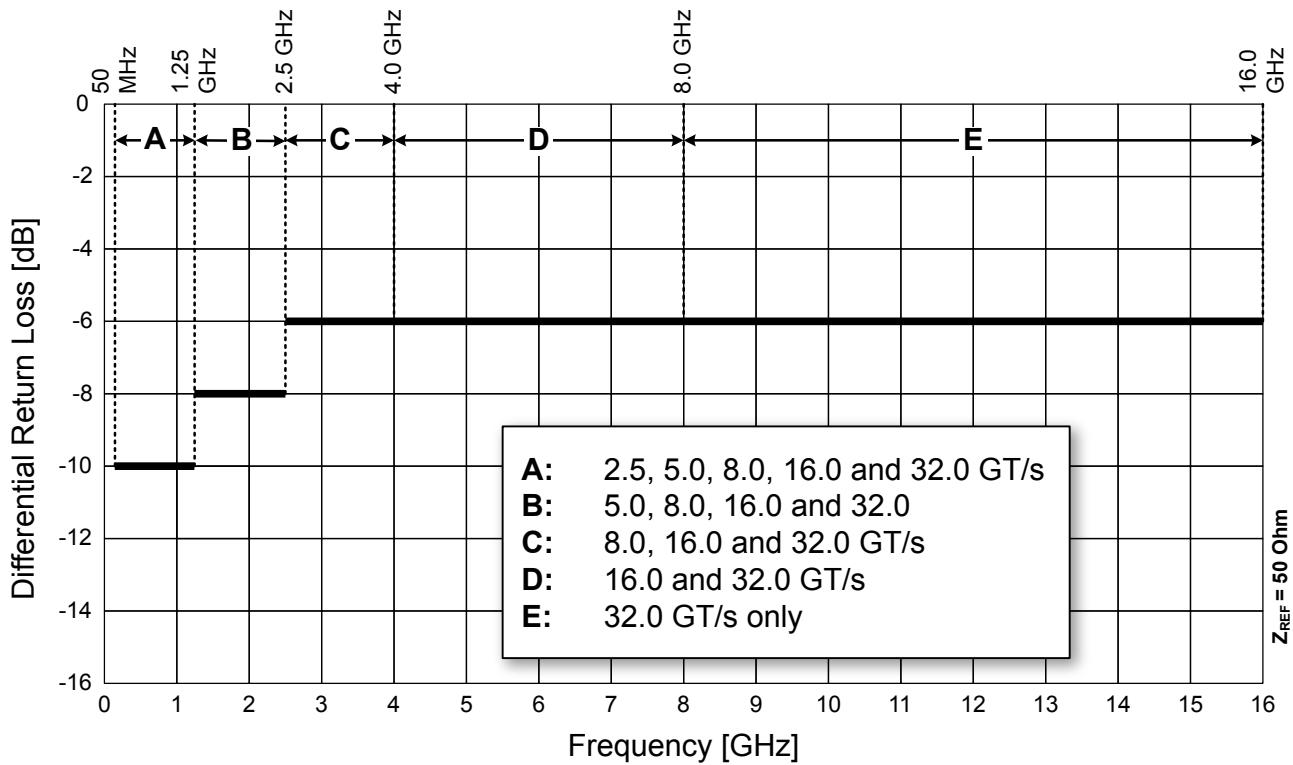


Figure 8-24 Tx, Rx Differential Return Loss Mask with 50 Ohm Reference §

The pass/fail mask for common mode return loss is shown in § Figure 8-25 for 2.5, 5.0, 8.0, 16.0, and 32.0 GT/s. Return loss measurements require that both the Tx and Rx are powered up and that their respective termination circuits are enabled.

Microprobing the package may be required to measure RL accurately.

Base 6.4 vs Base 6.3

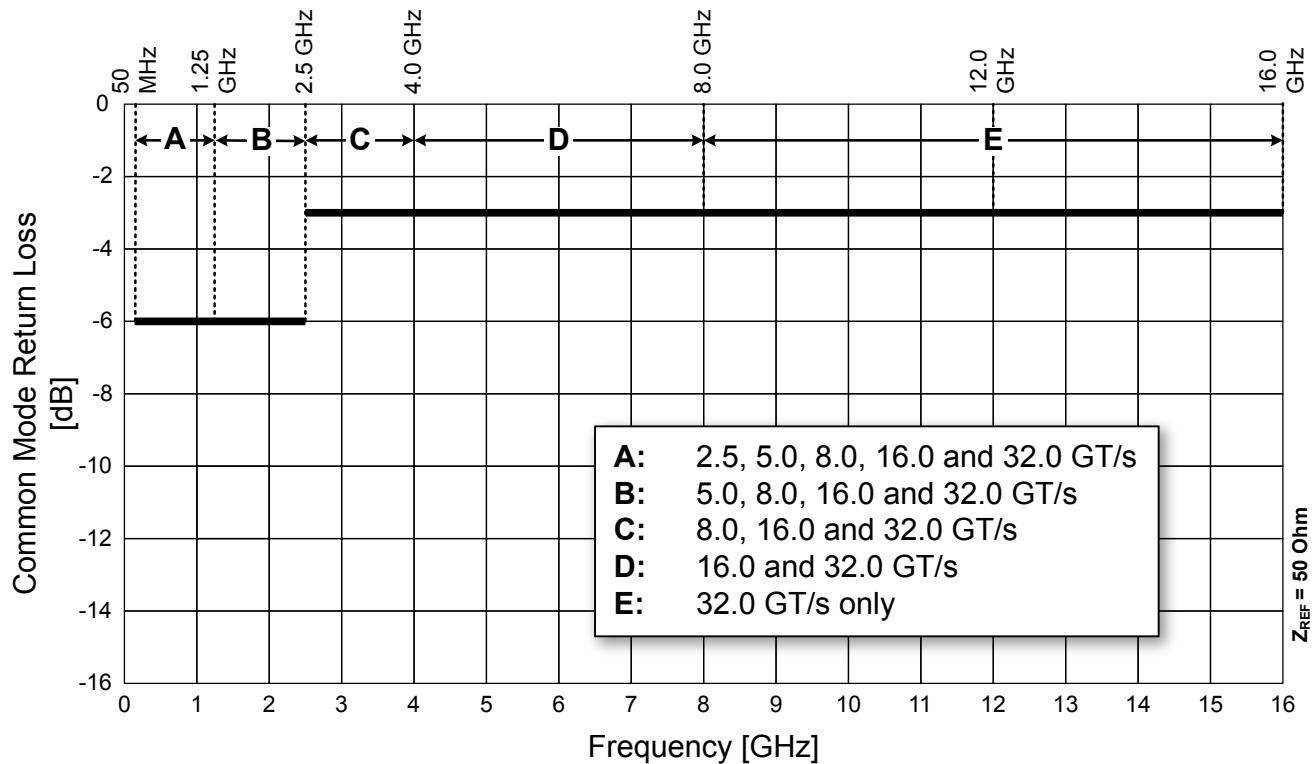


Figure 8-25 Tx, Rx Common Mode Return Loss Mask with 50 Ohm Reference §

8.3.8 Tx and Rx Return Loss for 64.0 GT/s §

§ Figure 8-26 defines the pass/fail mask for differential return loss for 64.0 GT/s with a single-ended reference impedance of 50 ohms.

Base 6.4 vs Base 6.3

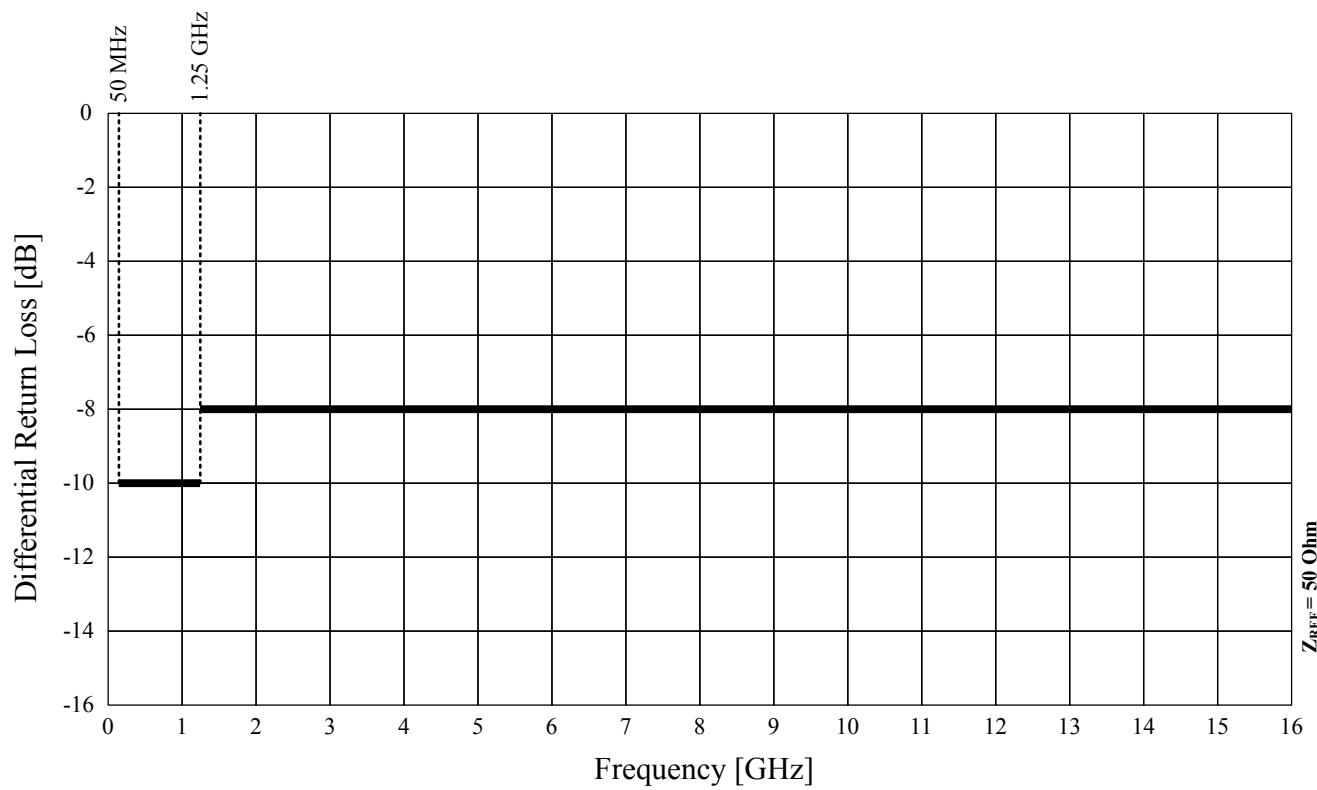


Figure 8-26 64.0 GT/s Tx, Rx Differential Return Loss Mask with 50 Ohm Reference §

§ Figure 8-27 defines the pass/fail mask for 64.0 GT/s common mode return loss with a single-ended reference impedance of 50 ohms. Return loss measurements require that both the Tx and Rx are powered up and that their respective termination circuits are enabled. Microprobing the package may be required to measure RL accurately.

Base 6.4 vs Base 6.3

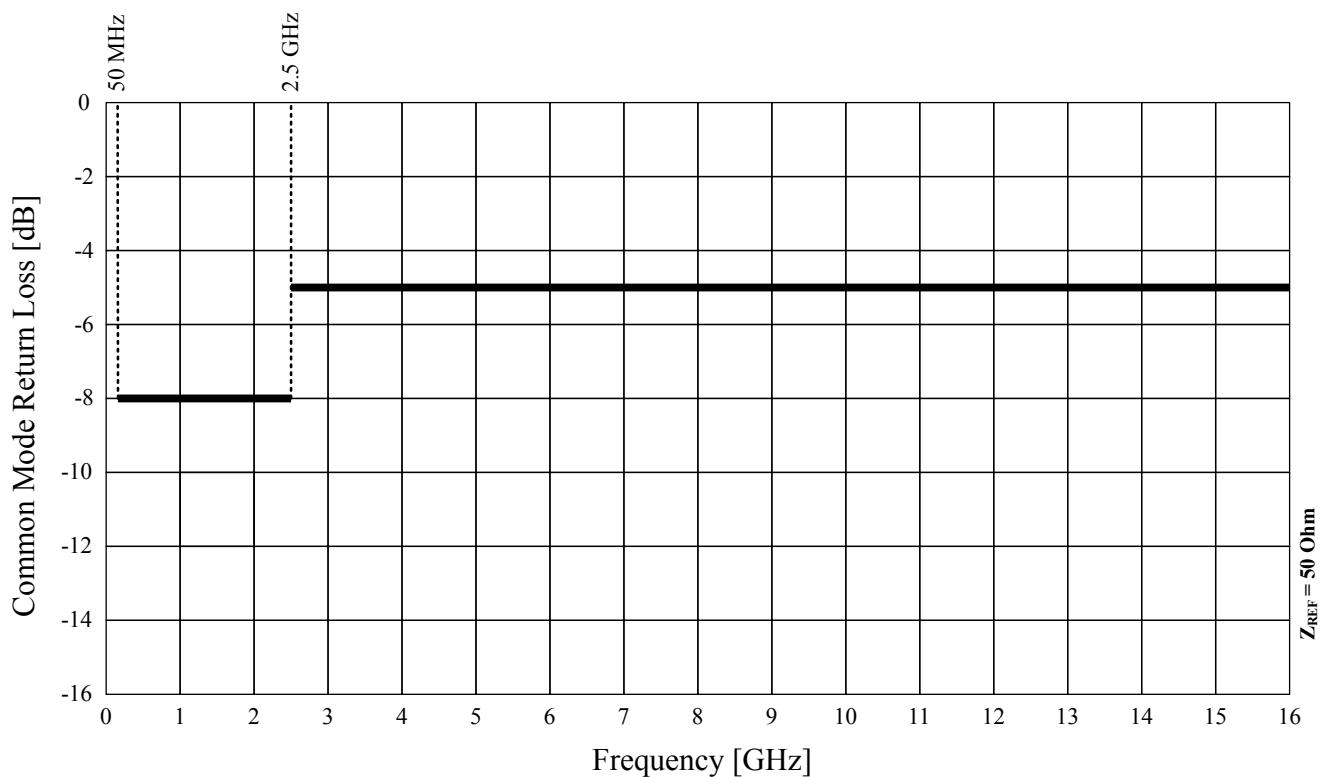


Figure 8-27 64.0 GT/s Tx, Rx Common Mode Return Loss Mask with 50 Ohm Reference §

8.3.9 Transmitter PLL Bandwidth and Peaking §

8.3.9.1 2.5 GT/s and 5.0 GT/s Tx PLL Bandwidth and Peaking §

PLL bandwidth and peaking are defined for both the Transmitter and Receiver in order to place an upper limit on the amount of Refclk jitter that is propagated to the transmitted data and to the CDR. Defining PLL BW and peaking limits also guarantees a minimum degree of Tx/Rx jitter tracking in those systems utilizing a Common Refclk Rx architecture.

The 2.5 GT/s PLL characteristics have been moved from the 3.0 CEM spec to the Electrical Base Spec. A single PLL bandwidth range from 1.5 to 22 MHz is given, which is identical to that defined in the CEM spec. No range of peaking was given in the CEM spec for the 2.5 GT/s PLL. However, for the Electrical Base Spec a peaking range of 0.01 dB to 3 dB is now defined. It is necessary to place a non-zero lower limit on the peaking, both to define a corner case as well as to maintain a common mathematical expression for the PLL transfer function in terms of ω_n and ζ .

Two sets of bandwidth and peaking are defined for 5.0 GT/s: 8-16 MHz with 3 dB of peaking and 5.0-16.0 MHz with 1 dB of peaking. This gives the designer the option of trading off between a low peaking PLL design vs. a low bandwidth design.

8.3.9.2 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s Tx PLL Bandwidth and Peaking §

The Tx and Rx PLL bandwidth for 8.0 and 16.0 GT/s is 0.5-5 MHz with 1.0 dB of peaking or 0.5-4 MHz with 2.0 dB of peaking. The Tx and Rx PLL bandwidth for 32.0 GT/s is 0.5 to 1.8 MHz with 2.0 dB of peaking. The Tx and Rx PLL bandwidth for 64.0 GT/s is 0.5 to 1.0 MHz with 2.0 dB of peaking. The 8.0 GT/s PLL BW range is substantially lower than

the PLL bandwidths specified for 5.0 GT/s or 2.5 GT/s to reduce the amount of Refclk jitter at the sample latch of the Receiver. A non-zero value of 0.01 dB is given for the lower limit of the peaking to define all the peaking corners.

8.3.9.3 Series Capacitors §

PCI Express requires series capacitors to provide a DC block between Tx and Rx. The min/max capacitance spread has been decreased from that of the 2.5 and 5.0 GT/s standards, while the maximum value has been slightly increased. This change is necessary to minimize DC wander effects due to data scrambling implemented at 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s. Note that 2.5 GT/s and 5.0 GT/s signaling must also propagate through these larger value capacitors, but the small increase in capacitor size has no adverse impact on either 2.5 GT/s or 5.0 GT/s signaling or low frequency in-band signaling such as Receiver detect.

8.3.10 Data Rate Independent Tx Parameters §

Table 8-7 Data Rate Independent Tx Parameters §

| Symbol | Parameter Description | Value | Units | Notes |
|----------------------------------|--|----------------------------|-------|--|
| $V_{TX-DC-CM}$ | Tx DC peak-peak common mode voltage | (min) 0 (max) 3.6 | V | Total single-ended voltage a Tx can supply under any conditions with respect to ground. See also the $I_{TX-SHORT}$. See Note 1 |
| $V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$ | Absolute delta of DC Common Mode Voltage during L0 and Electrical Idle | (min) 0 (max) 100 | mV | $ V_{TX-CM-DC} \text{ [during L0]} - V_{TX-CM-Idle-DC} \text{ [during Electrical Idle]} \leq 100 \text{ mV}$ $V_{TX-CM-DC} = \text{DC (avg)} \text{ of } V_{TX-D+} + V_{TX-D-} / 2$ $V_{TX-CM-Idle-DC} = \text{DC (avg)} \text{ of } V_{TX-D+} + V_{TX-D-} / 2 \text{ [Electrical Idle]}$ |
| $V_{TX-CM-DC-LINE-DELTA}$ | Absolute Delta of DC Common Mode Voltage between D+ and D- | (min) 0 (max) 25 | mV | $ V_{TX-CM-DC-D+} \text{ [during L0]} - V_{TX-CM-DC-D-} \text{ [during L0]} \leq 25 \text{ mV}$ $V_{TX-CM-DC-D+} = \text{DC (avg)} \text{ of } V_{TX-D+} \text{ [during L0]} $ $V_{TX-CM-DC-D-} = \text{DC (avg)} \text{ of } V_{TX-D-} \text{ [during L0]} $ |
| $V_{TX-IDLE-DIFF-AC-p}$ | Electrical Idle Differential Peak Output Voltage | (min) 0 (max) 20 | mV | $V_{TX-IDLE-DIFF-AC-p} = V_{Tx-Idle-D+} - V_{Tx-Idle-D-} \leq 20 \text{ mV}$. Voltage must be band pass filtered to remove any DC component and HF noise. The bandpass is constructed from two first-order filters, the high pass and low pass 3 dB bandwidths are 10 kHz and 1.25 GHz, respectively. |
| $V_{TX-IDLE-DIFF-DC}$ | DC Electrical Idle Differential Output Voltage | (min) 0 (max) 5 | mV | $V_{TX-IDLE-DIFF-DC} = V_{Tx-Idle-D+} - V_{Tx-Idle-D-} \leq 5 \text{ mV}$. Voltage must be low pass filtered to remove any AC component. The low pass filter is first-order with a 3 dB bandwidth of 10 kHz. |
| $V_{TX-RCV-DETECT}$ | The amount of voltage change allowed during Receiver Detection | (max) 600 | mV | The total amount of voltage change in a positive direction that a Transmitter can apply to sense whether a low impedance Receiver is present. Note: Receivers display substantially different impedance for $V_{IN} < 0$ vs. $V_{IN} > 0$. |

| Symbol | Parameter Description | Value | Units | Notes |
|----------------------------|--|------------------------|-----------------|---|
| $T_{TX-IDLE-MIN}$ | Minimum time spent in Electrical Idle | 20 (min) | ns | The time a Tx must spend in Electrical Idle before transitioning to another state |
| $T_{TX-IDLE-SET-TO-IDLE}$ | Maximum time to transition to a valid Electrical Idle after sending an EIOS | (max) 8 | ns | After sending the required number of EIOSs, the Transmitter must meet all Electrical Idle specifications within this time. This is measured from the end of the last UI of the last EIOS to the Transmitter in Electrical Idle. |
| $T_{TX-IDLE-TO-DIFF-DATA}$ | Maximum time to transition to valid diff signaling after leaving Electrical Idle | (max) 8 | ns | Maximum time to transition to valid diff signaling after leaving Electrical Idle. This is considered a debounce time to the Tx. |
| $T_{CROSSLINK}$ | Crosslink random timeout | (max) 1.0 | ms | This random timeout helps resolve potential conflicts in the crosslink configuration. |
| C_{TX} | AC Coupling Capacitor | (min) 176 (max) 265 | nF | All Transmitters shall be AC coupled. The AC coupling is required either within the media or within the transmitting component itself. |
| $Z_{TX-DIFF-DC}$ | DC differential Tx impedance | (max) 120 | Ω | Low impedance defined during signaling. The minimum value is bounded by $R_{L,TX-DIFF}$. |
| $I_{TX-SHORT}$ | Tx short circuit current | (max) 90 | Normal ↑↑mA↑ | Tx short circuit current. Note 1 |

Notes:

1. $I_{TX-SHORT}$ and $V_{TX-DC-CM}$ stipulate the maximum current/voltage levels that a Transmitter can generate and therefore define the worst case transients that a Receiver must tolerate.

8.4 Receiver Specifications §

8.4.1 Receiver Stressed Eye Specification §

All Receiver speeds are tested by means of a stressed eye applied over a calibration channel that approximates the near worst-case loss characteristics encountered in an actual channel. The recovered eye is defined at the input to the Receiver's latch. For 2.5 GT/s and 5.0 GT/s this point is equivalent to the Rx pins; for 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s it is equivalent to the signal at the Rx die pad after behavioral Rx equalization has been applied.

8.4.1.1 Breakout and Replica Channels §

The closest practical measurement points to the Rx DUT are the coaxial connectors at the end of a breakout channel, while the Rx reference point of interest is the pin of the Rx. By constructing a replica channel that closely matches the electrical characteristics of the breakout channel it is possible to measure the signal as it would appear at the DUT's pin, if the DUT were an ideal termination. Impedance targets for the Rx breakout and replica channels are $85\ \Omega$ differential and $42.5\ \Omega$ single-ended, and the impedance tolerance should be maintained within $\pm 5\%$ or better. Note that the impedance target for the Tx test breakout and replica channels is still $100\ \Omega$ differential and $50\ \Omega$ single-ended.

In § Figure 8-28 the stressed eye is observed at TP2 with the signal sources connected to the calibration channel. A calibration channel will be required for each data rate. Once the stressed eye has been calibrated, the signal source is applied to the DUT. Note that TP1-TP2 encompasses all the components between the signal source and the equivalent of the DUT pin, thereby capturing all non-ideal characteristics in the overall insertion loss due to cabling and replica/breakout channel, excluding Rx package. The AC and DC loss from generator to TP1 are assumed to be zero or must be otherwise de-embedded. The $V_{RX-LAUNCH}$ (differential voltage swing) and Tx Equalization of the Signal Generator are calibrated at TP3 as shown in § Figure 8-28. Some Signal Generators do factory calibration with a cable and trying to de-embed to TP1 for these calibrations can cause inaccuracies. Only the loss from TP3 onward is counted in overall calibration channel loss.

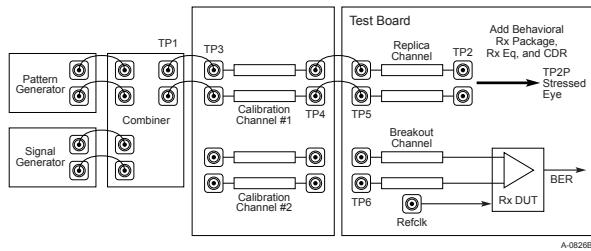


Figure 8-28 Rx Test board Topology for 16.0 and 32.0 GT/s §

8.4.1.2 Calibration Channel Insertion Loss Characteristics §

Calibration channels, each with a specified differential insertion loss at one of the PCIe data rates, provide the means of generating prescribed amounts of ISI that approximates a worst-case channel. For each data rate a single calibration channel loss mask is defined by means of two pairs of IL limits at a high and a low frequency. It is not acceptable to generate IL by means other than physical channel (PCB traces, cables, switches, small compensation delays, etc. are acceptable), such as specialized filters. The Calibration Channel needs to include all physical loss after TP3 as shown in § Figure 8-28 within the IL mask.

Base 6.4 vs Base 6.3

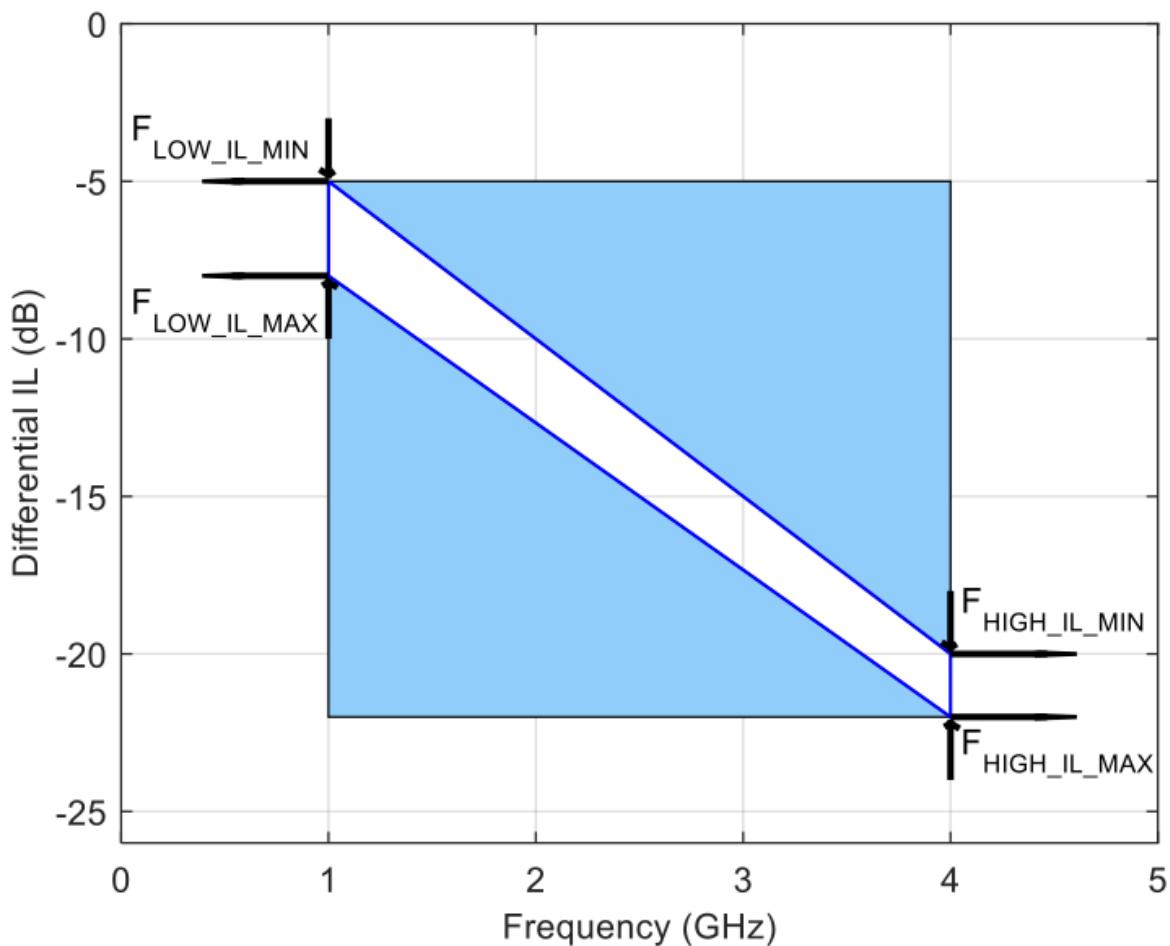


Figure 8-29 Example Calibration Channel IL Mask Excluding Rx Package for 8.0 GT/s §

The following table defines the calibration channel IL masks by means of four loss points at two frequency limits, thereby creating a quadrilateral shaped solution area. The calibration channel IL mask is provided as a guideline to minimize reflection so that the channel represents mostly ISI stress for the receiver test.

Table 8-8 Calibration Channel IL Limits §

| Data Rate | $F_{LOW-IL-MIN}$ | $F_{LOW-IL-MAX}$ | $F_{HIGH-IL-MIN}$ | $F_{HIGH-IL-MAX}$ |
|-------------------------|------------------|------------------|-------------------|-------------------|
| 2.5 GT/s | 4.5 dB @ 1 GHz | 5.0 dB @ 1 GHz | 4.7 dB @ 1.25 GHz | 5.2 dB @ 1.25 GHz |
| 5.0 GT/s | 4.5 dB @ 1 GHz | 5.0 dB @ 1 GHz | 10.0 dB @ 2.5 GHz | 11.0 dB @ 2.5 GHz |
| 8.0 GT/s | 5 dB @ 1 GHz | 8 dB @ 1 GHz | 20 dB @ 4 GHz | 22 dB @ 4 GHz |
| 16.0 GT/s Root Port | 4.2 dB @ 1 GHz | 5.2 dB @ 1 GHz | 22.5 dB @ 8 GHz | 23.5 dB @ 8 GHz |
| 16.0 GT/s Non-Root Port | 4.2 dB @ 1 GHz | 5.2 dB @ 1 GHz | 24.5 dB @ 8 GHz | 25.5 dB @ 8 GHz |
| 32.0 GT/s Root Port | 3.2 dB @ 1 GHz | 4.2 dB @ 1 GHz | 26.5 dB @ 16 GHz | 27.5 dB @ 16 GHz |
| 32.0 GT/s Non-Root Port | 3.9 dB @ 1 GHz | 4.9 dB @ 1 GHz | 31.5 dB @ 16 GHz | 32.5 dB @ 16 GHz |

| Data Rate | $F_{LOW-IL-MIN}$ | $F_{LOW-IL-MAX}$ | $F_{HIGH-IL-MIN}$ | $F_{HIGH-IL-MAX}$ |
|-------------------------|------------------|------------------|-------------------|-------------------|
| 64.0 GT/s Root Port | 3.0 dB @ 1 GHz | 4.0 dB @ 1 GHz | 23.5 dB @ 16 GHz | 24.5 dB @ 16 GHz |
| 64.0 GT/s Non-Root Port | 3.6 dB @ 1 GHz | 4.6 dB @ 1 GHz | 27.5 dB @ 16 GHz | 28.5 dB @ 16 GHz |

Notes:

- Calibration channel plus Rx package is 28 dB nominally (informative) for 16.0 GT/s.
- Calibration channel plus Rx package is 36 dB nominally (informative) for 32.0 GT/s.
- Calibration channel plus Rx package is 32 dB nominally (informative) for 64.0 GT/s.
- Different reference packages are defined for devices containing Root Ports and all other device types at 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s.
- It is recommended that some validation be done with shorter channels at 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s.
- For 32.0 GT/s, a material at least as good as a Megtron-6 class material with loss of approximately 1.0 dB/inch at 16 GHz at typical room conditions must be used.
- For 64.0 GT/s, a material at least as good as a Megtron-6 class material with loss of approximately 1.0 dB/inch at 16 GHz under worst-case temperature and humidity conditions must be used to achieve system routing length of 13" for 1-connector server topologies.

The impedance targets for the Rx tolerancing interconnect environment are $100\ \Omega$ differential and $50\ \Omega$ single-ended for the 2.5, 5.0, and 8.0 GT/s channels and $85\ \Omega$ differential and $42.5\ \Omega$ single-ended for the 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s channels; the impedance tolerance should be maintained within $\pm 5\%$ or better.

The calibration channel for 16.0 GT/s must meet the following return loss mask when measured from either end of the calibration channel:

- $\leq -12\ dB$ for $< 4\ GHz$
- $\leq -8\ dB$ for $\geq 4\ GHz$ and $< 12\ GHz$
- $\leq -6\ dB$ for $\geq 12\ GHz$ and $\leq 16\ GHz$

The calibration channel for 32.0 GT/s and 64.0 GT/s must meet the following return loss mask when measured from either end of the calibration channel:

- $\leq -12\ dB$ for $< 4\ GHz$
- $\leq -10\ dB$ for $\geq 4\ GHz$ and $< 16\ GHz$
- $\leq -6\ dB$ for $\geq 16\ GHz$ and $\leq 32\ GHz$

A calibration channel consists of a differential pair of PCB traces terminated at both ends by coaxial connectors. For 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s the calibration channel includes a 4.0 (16.0 GT/s), 5.0 (32.0 GT/s), or 6.0 (64.0 GT/s) Card Electromechanical Specification compliant connector and edge finger placed at least 4 dB at Nyquist away from the coaxial connectors where the signal generator is connected. The calibration channel's electrical characteristics are defined in terms of differential insertion loss masks as shown in § Figure 8-29 , where S_{DD21} is measured between TP3 (See § Figure 8-28) and TP2. Connections between TP4-TP5 represent cabling and are included in the S_{DD21} measurement. Loss before TP3 is effectively calibrated out by calibrating differential voltage swing and TX EQ at TP3 and is not included in the S_{DD21} measurement.

While the 8.0 GT/s, 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s S-parameter masks do not extend below 1.0 GHz, all calibration channels must be well behaved below 1.0 GHz and must not have a DC resistance in excess of 7.5 ohms, as measured by the sum of the resistances of the D+ and D- traces. This limitation on DC resistance guarantees that the calibration channel low frequency characteristic is consistent with the extrapolations of the S_{DD21} masks to DC. The calibration loss targets for devices containing Root Ports and other devices are different because the reference package models have

different losses. For 16.0 GT/s, 32.0 GT/s, and 64.0 GT/s the insertion loss range $F_{HIGH-IL-MIN}$ to $F_{HIGH-IL-MAX}$ is the nominal loss. The calibration channel must have a series of loss options covering a range from at least 2 dB below $F_{HIGH-IL-MIN}$ to 3 dB above $F_{HIGH-IL-MAX}$ (for example for the non-root case this means a loss range from -22.5 to -28.5 dB) with loss delta between consecutive options of 0.5 dB or less.

IMPLEMENTATION NOTE:

16.0 GT/S CALIBRATION CHANNEL REFERENCE DESIGN §

This section gives an example of a 16.0 GT/s calibration channel that was built and tested to meet the requirements in this specification. A high-level block diagram of the calibration channel is shown in § Figure 8-30 . Note that this example fixture covers a wider loss range than required by the specification and can cover both root and non-root cases. The test fixture includes four PCBs:

16.0 GT/s Rx Calibration Base Boards

Sixteen differential pairs (85 Ohm Nominal Impedance) routed from SMA connectors to a CEM through-hole connector. There are three different base boards to achieve the following insertion loss ranges – The insertion loss of the differential pairs for the base board is varied as follows @ 8.0 GHz in 0.5 dB steps.

Low-Loss Base Board: 4-11.5 dB

Mid-Loss Base Board: 12-19.5 dB

High-Loss Base Board: 20-27.5 dB

All traces are routed as microstrip on the bottom layer. The SMA connectors and CEM connectors are optimized with layout techniques at 8.0 GHz.

16.0 GT/s Rx Calibration Riser Board

Sixteen differential pairs (85 Ohm Nominal Impedance) routed from SMA connectors to Gold Edge Fingers. The insertion loss of the differential pairs is fixed at 4 dB nominal @ 8.0 GHz for all sixteen pairs. All traces are routed as microstrip on the bottom layer. The SMA connectors and CEM connectors are optimized with layout techniques at 8.0 GHz.

Base 6.4 vs Base 6.3

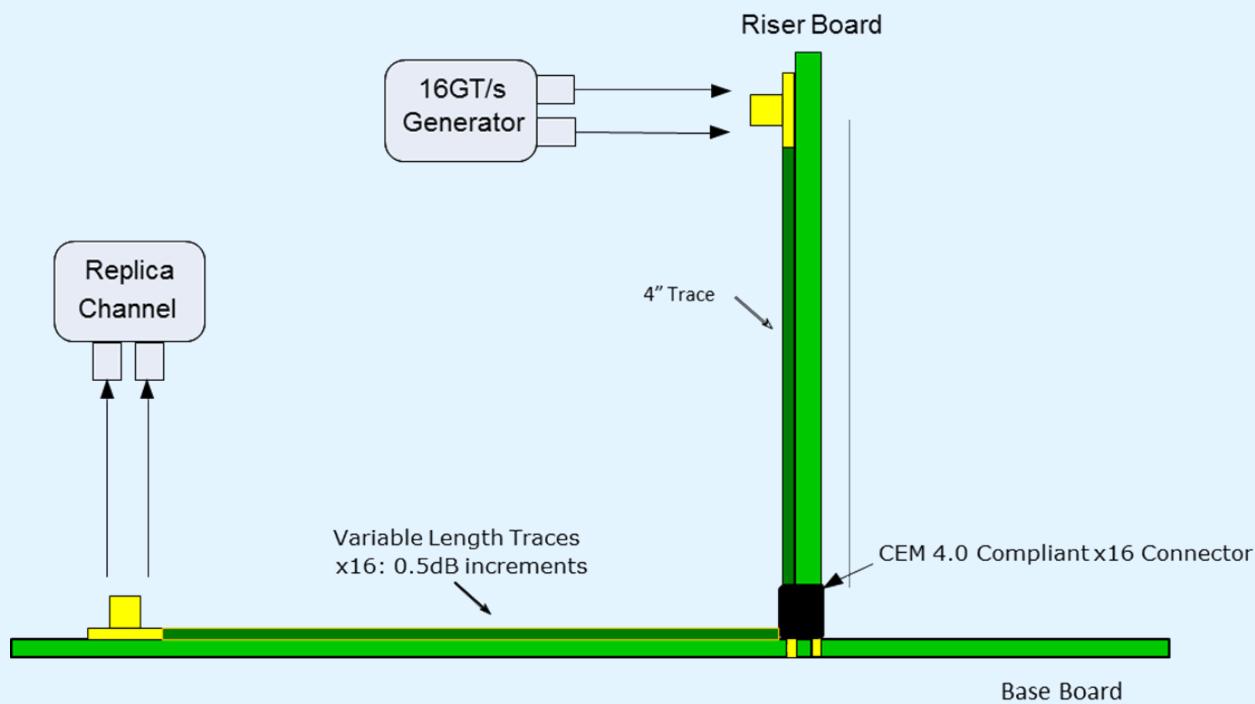


Figure 8-30 Example 16.0 GT/s Calibration Channel §

The stackup for both boards is shown in § Figure 8-31 where 65% is the estimated copper fill percentage. § Figure 8-31 includes stackups for both nominal $85\ \Omega$ and $100\ \Omega$ stackups - the $85\ \Omega$ stackup is used for the calibration channel example.

Figure 8-31 Stackup for Example 16.0 GT/s Calibration Channel §

The pad stack for the CEM connector drill holes is shown in § Figure 8-32 and the pad stack for the SMA drill holes is shown in § Figure 8-33 .

Base 6.4 vs Base 6.3

Padstack: C40P28P3M3-A59 Type: through Inner pads: Optional

| Layer | Pad Type | Geometry | Width | Height | Offset X | Offset Y | Flash Name | Shape Name |
|-------------------|----------|-----------|-------|--------|----------|----------|-----------------|------------|
| TOP | ANTI | CIRCLE | 59.00 | 59.00 | 0.00 | 0.00 | | |
| TOP | THERMAL | RECTANGLE | 80.00 | 80.00 | 0.00 | 0.00 | TH80X60X15_4X45 | |
| TOP | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| GND-2 | ANTI | CIRCLE | 59.00 | 59.00 | 0.00 | 0.00 | | |
| GND-2 | THERMAL | RECTANGLE | 80.00 | 80.00 | 0.00 | 0.00 | TH80X60X15_4X45 | |
| GND-2 | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| GND-3 | ANTI | CIRCLE | 59.00 | 59.00 | 0.00 | 0.00 | | |
| GND-3 | THERMAL | RECTANGLE | 80.00 | 80.00 | 0.00 | 0.00 | TH80X60X15_4X45 | |
| GND-3 | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| BOTTOM | ANTI | CIRCLE | 59.00 | 59.00 | 0.00 | 0.00 | | |
| BOTTOM | THERMAL | RECTANGLE | 80.00 | 80.00 | 0.00 | 0.00 | TH80X60X15_4X45 | |
| BOTTOM | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| internal_pad_def | ANTI | CIRCLE | 59.00 | 59.00 | 0.00 | 0.00 | | |
| internal_pad_def | THERMAL | RECTANGLE | 80.00 | 80.00 | 0.00 | 0.00 | TH80X60X15_4X45 | |
| internal_pad_def | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| SOLDERMASK_TOP | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| SOLDERMASK_BOTTOM | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| PASTEMASK_TOP | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |
| PASTEMASK_BOTTOM | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |
| FILMMASKTOP | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |
| FILMMASKBOTTOM | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |

Drill Data for C40P28P3M3-A59

| Hole Type | Drill Dia | Plating | Figure | Characters | Width | Height | Offset X | Offset Y | Pos Tolerance | Neg Tolerance | Non-Standard |
|--------------|-----------|---------|---------|------------|-------|--------|----------|----------|---------------|---------------|--------------|
| CIRCLE DRILL | 28.00 | PLATED | DIAMOND | | 50.00 | 50.00 | 0.00 | 0.00 | 3.00 | 3.00 | |

Figure 8-32 CEM Connector Drill Hole Pad Stack §

Base 6.4 vs Base 6.3

Padstack: C60_BOT75P20_SSM40P3M3 Type: through Inner pads: Optional

| Layer | Pad Type | Geometry | Width | Height | Offset X | Offset Y | Flash Name | Shape Name |
|-------------------|----------|-----------|-------|--------|----------|----------|------------|------------|
| TOP | ANTI | CIRCLE | 45.00 | 45.00 | 0.00 | 0.00 | | |
| TOP | THERMAL | RECTANGLE | 5.00 | 5.00 | 0.00 | 0.00 | 5MIL_PAD | |
| TOP | REGULAR | CIRCLE | 60.00 | 60.00 | 0.00 | 0.00 | | |
| GND-2 | ANTI | CIRCLE | 45.00 | 45.00 | 0.00 | 0.00 | | |
| GND-2 | THERMAL | RECTANGLE | 5.00 | 5.00 | 0.00 | 0.00 | 5MIL_PAD | |
| GND-2 | REGULAR | CIRCLE | 60.00 | 60.00 | 0.00 | 0.00 | | |
| GND-3 | ANTI | CIRCLE | 45.00 | 45.00 | 0.00 | 0.00 | | |
| GND-3 | THERMAL | RECTANGLE | 5.00 | 5.00 | 0.00 | 0.00 | 5MIL_PAD | |
| GND-3 | REGULAR | CIRCLE | 60.00 | 60.00 | 0.00 | 0.00 | | |
| BOTTOM | ANTI | CIRCLE | 45.00 | 45.00 | 0.00 | 0.00 | | |
| BOTTOM | THERMAL | RECTANGLE | 5.00 | 5.00 | 0.00 | 0.00 | 5MIL_PAD | |
| BOTTOM | REGULAR | CIRCLE | 75.00 | 75.00 | 0.00 | 0.00 | | |
| internal_pad_def | ANTI | CIRCLE | 45.00 | 45.00 | 0.00 | 0.00 | | |
| internal_pad_def | THERMAL | RECTANGLE | 5.00 | 5.00 | 0.00 | 0.00 | 5MIL_PAD | |
| internal_pad_def | REGULAR | CIRCLE | 60.00 | 60.00 | 0.00 | 0.00 | | |
| SOLDERMASK_TOP | REGULAR | CIRCLE | 60.00 | 60.00 | 0.00 | 0.00 | | |
| SOLDERMASK_BOTTOM | REGULAR | CIRCLE | 40.00 | 40.00 | 0.00 | 0.00 | | |
| PASTEMASK_TOP | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |
| PASTEMASK_BOTTOM | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |
| FILMMASKTOP | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |
| FILMMASKBOTTOM | REGULAR | NULL | 0.00 | 0.00 | 0.00 | 0.00 | | |

Drill Data for C60_BOT75P20_SSM40P3M3

| Hole Type | Drill Dia | Plating | Figure | Characters | Width | Height | Offset X | Offset Y | Pos Tolerance | Neg Tolerance | Non-Standard |
|--------------|-----------|---------|--------|------------|-------|--------|----------|----------|---------------|---------------|--------------|
| CIRCLE DRILL | 20.00 | PLATED | CIRCLE | E | 60.00 | 60.00 | 0.00 | 0.00 | 3.00 | 3.00 | |

Figure 8-33 Pad Stack for SMA Drill Holes §

IMPLEMENTATION NOTE:

32.0 GT/S CALIBRATION CHANNEL REFERENCE DESIGN §

This section gives an example of a 32.0 GT/s calibration channel that was built and tested to meet the requirements in this specification. A high-level block diagram of the calibration channel is shown in § Figure 8-34 . Note this example fixture covers a wider loss range then required by the specification and can cover both root and non-root cases. The test fixture includes two PCBs:

32.0 GT/s Rx Calibration Base Boards

Sixteen differential pairs (85 Ohm Nominal Impedance) routed on a Megtron-6 PCB from MMPX connectors to a CEM surface mount connector. There are three different base boards to achieve the following insertion loss ranges - The insertion loss of the differential pairs for the base board is varied as follows @ 16.0 GHz in 0.5 dB steps.

Low-Loss Base Board: 4.0-11.5 dB

Mid-Loss Base Board: 12.0-19.5 dB

High-Loss Base Board: 20.0-27.5 dB

All traces are routed as microstrip on the top layer. The MMPX connectors and CEM connectors are optimized with layout techniques at 16.0 GHz

For information on MMPX connectors refer to Huber+Suhner microminiature connectors.

32.0 GT/s Rx Calibration Riser Board

Sixteen differential pairs (85 Ohm Nominal Impedance) routed on a Megtron-6 PCB from MMPX connectors to Gold Edge Fingers. The insertion loss of the differential pairs is fixed at 8 dB nominal @ 16.0 GHz for all sixteen pairs. All traces are routed as microstrip on the top layer. The MMPX connectors and CEM connectors are optimized with layout techniques at 16.0 GHz.

Base 6.4 vs Base 6.3

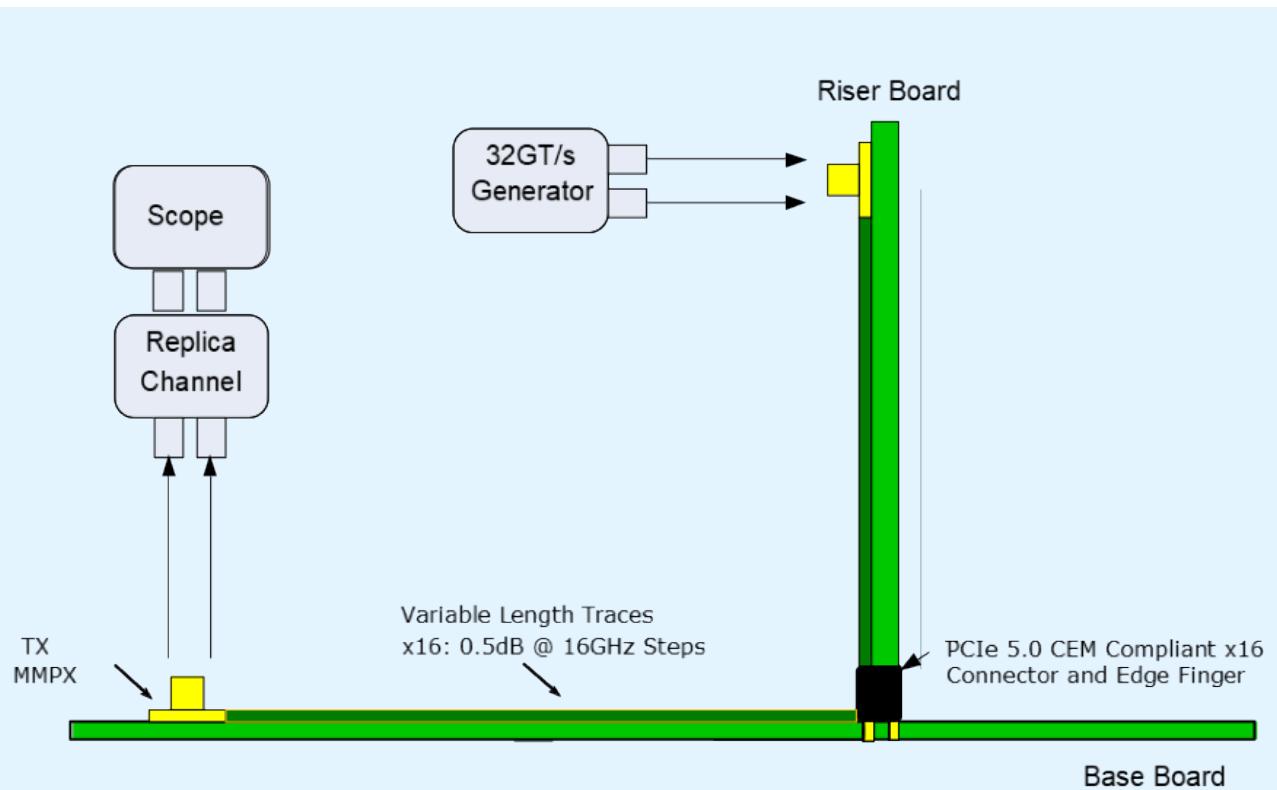


Figure 8-34 Example 32.0 GT/s Calibration Channel §

The stack-up for both 85 Ohm boards is shown in § Figure 8-35 where 65% is the estimated copper fill percentage.

| Material | Layer | Starting Cu oz | Finished Cu oz | Dielectric | Single Ended Impedance Lyr | Designed Trace Width | Finished Trace Width | Calculated Impedance Lyr | Ref Lyr A | Ref Lyr B | Differential Impedance | Designed Trace/Space | Finished Trace/Space | Calculated Impedance Lyr | Ref Lyr A | Ref Lyr B |
|--|-------|----------------|----------------|------------|----------------------------|----------------------|----------------------|--------------------------|---------------|----------------------------|--------------------------|----------------------|----------------------|--------------------------|-----------|-----------|
| 1-3313(54%), 1-1078(75%) | 1 | Signal .5 | 1.5 | .0065 | 42.5 Ω 50 Ω | .00945 | .0155 | 42.18 Ω 49.7 Ω | 85 Ω 100 Ω | .00675/.00525 .005/.007 | .0115/.0095 .006/.006 | 85.94 Ω 100.45 Ω | 2 | 2 | | |
| Filler Material | 2 | Plane 1 | 1 | .042 | | | | | | | | | | | | |
| 1-3313(54%), 1-1078(75%) | 3 | Plane 1 | 1 | .0065 | | | | | | | | | | | | |
| | 4 | Signal .5 | 1.5 | .0065 | 50 Ω 42.5 Ω | .008 .00945 | .0115 .0155 | 49.7 Ω 42.18 Ω | 100 Ω 85 Ω | .005/.007 .00675/.00525 | .006/.006 .0115/.0095 | 100.45 Ω 85.94 Ω | 2 | 2 | | |
| Final Thickness (After Plating): 0.062 | | | | | | | | | | | | | | | | |

Figure 8-35 Stack-up for Example 32.0 GT/s Calibration Channel §

64.0 GT/s Calibration channel reference design may be added in Rev 0.9.

8.4.1.3 Post Processing Procedures §

The Receiver test requires that the stressed eye characteristics be measured at TP2 (which is accessible) and then post-processed to yield a signal as it would appear at test point two post-processed (TP2P) (which is not accessible) for 8.0, 16.0, 32.0, and 64.0 GT/s. TP2P defines a reference point that comprehends the effects of the behavioral Rx package plus Rx equalization and represents the only location where a meaningful EH and EW limits can be defined.

8.4.1.4 Behavioral Rx Package Models §

Behavioral Rx package models are included as part of the post processing to allow the calibrated eye to comprehend package insertion loss. A separate pair of package models is defined for 8.0, 16.0, 32.0, and 64.0 GT/s eye calibration. At 8.0 GT/s, separate package models are defined for TX and RX ports to reflect the smaller CPAD capacitance typical in most receiver implementations. At 16.0, 32.0, and 64.0 GT/s, separate package models are defined for devices containing Root Ports and all other devices. This is necessary to allow a reasonable channel solution space and assumes that devices containing Root Complexes are usually large and socketed, while all other devices tend to be unsocketed and smaller. The 16.0 GT/s Root and Non-Root behavioral Rx package models have been constructed to represent respective package loss characteristics for high loss, but not worst-case loss, packages. The 32.0 and 64.0 GT/s Root and Non-Root behavioral Rx package models have been constructed to represent package loss characteristics for worst case packages.

The 8.0, 32.0, and 64.0 GT/s stressed eye test for all devices and the 16.0 GT/s stressed eye test for Non-Root Package devices that support captive channels are required to use the appropriate behavioral package (see § Section 8.3.3.11). For all other device types, if the actual Rx package performance is worse than that of the behavioral package, then the actual package models are permitted to be used. If the actual package models are used, the calibration channel must be adjusted such that the total channel loss including the embedded actual package remains at 28 dB nominal. Note that form factor overall requirements still need to be met. The Rx package performance is assessed using the methodology defined in § Section 8.5.1.2.

Details of the behavioral Rx packages are provided in § Section 8.5.1.1 of the Channel Tolerancing section. S-parameter models for the behavioral Rx package models are available as design collateral. The reference impedance at the pad side of the packages model is assumed to be $2 \times 50 \Omega$.

8.4.1.5 Behavioral CDR Model §

Post processing shall include a behavioral CDR model with a data rate dependent transfer function. A first order CDR transfer function is utilized for Receivers operating with a CC Refclk architecture except for 32.0 GT/s and 64.0 GT/s. For Receivers operating in IR Refclk mode an alternate CDR transfer function is required. For a given data rate the behavioral CDR used for Rx testing is the same as the corresponding CDR used for Tx testing. For details on behavioral CDR functions refer to § Section 8.3.5.5.

8.4.1.6 No Behavioral Rx Equalization for 2.5 and 5.0 GT/s §

The combination of worst-case channel, behavioral Rx package, and Tx jitter at 2.5 and 5.0 GT/s will yield open eyes, when the appropriate Tx presets are set. Therefore, there is no need to define a behavioral Rx equalization or to adjust the Tx equalization setting. Actual implementations of 2.5 and 5.0 GT/s receivers may, of course, include equalization.

8.4.1.7 Behavioral Rx Equalization for 8.0, 16.0, 32.0, and 64.0 GT/s §

As measured at TP2, stressed eyes at 8.0, 16.0, 32.0, and 64.0 GT/s will usually be closed, making direct measurement of the stressed eye jitter parameters unfeasible. This problem is overcome by employing a behavioral Receiver equalizer that implements both CTLE and a 1-tap DFE (8.0 GT/s) or a 2-tap DFE (16.0 GT/s) or a 3-tap DFE (32.0 GT/s) or a 16-tap DFE (64.0 GT/s).

Rx equalization algorithms of CTLE and DFE are only intended to be a means for obtaining an open eye in the presence of calibration channel ISI plus the other signal impairment terms and for channel compliance. The behavioral Rx equalization algorithms are not intended to serve as a guideline for implementing actual Receiver equalization. For example, additional DFE taps can have significant benefit in actual implementations where the CTLE may differ from the

behavioral equalizer and/or CTLE selection may not always be optimal. Channel loss characteristics can vary significantly with temperature and humidity and a real Receiver must be able to continue to function at the target BER through such variations.

8.4.1.8 Behavioral CTLE (8.0 and 16.0 GT/s) §

8.0 and 16.0 GT/s behavioral Rx equalization defines a 1st order CTLE with fixed LF and HF poles, and an adjustable DC gain (A_{DC}) specified according to the family of curves shown in § Figure 8-37 . For the 8.0 GT/s rates A_{DC} is adjustable over a minimum range of -6 to -12 dB in steps of 1.0 dB.

$$H(s) = \omega_{P2} \times \frac{s + \omega_{P1} \times A_{DC}}{(s + \omega_{P1}) \times (s + \omega_{P2})}$$

$$\omega_{P1} = \text{pole 1} = 2\pi \times 2 \text{ GHz}$$

$$\omega_{P2} = \text{pole 2} = 2\pi \times 8 \text{ GHz}$$

$$A_{DC} = \text{dc gain}$$

Figure 8-36 Transfer Function for 8.0 GT/s Behavioral CTLE §

The following diagram illustrates the gain vs. frequency behavior of the CTLE as A_{DC} is varied over its minimum to maximum range in 1.0 dB steps.

Base 6.4 vs Base 6.3

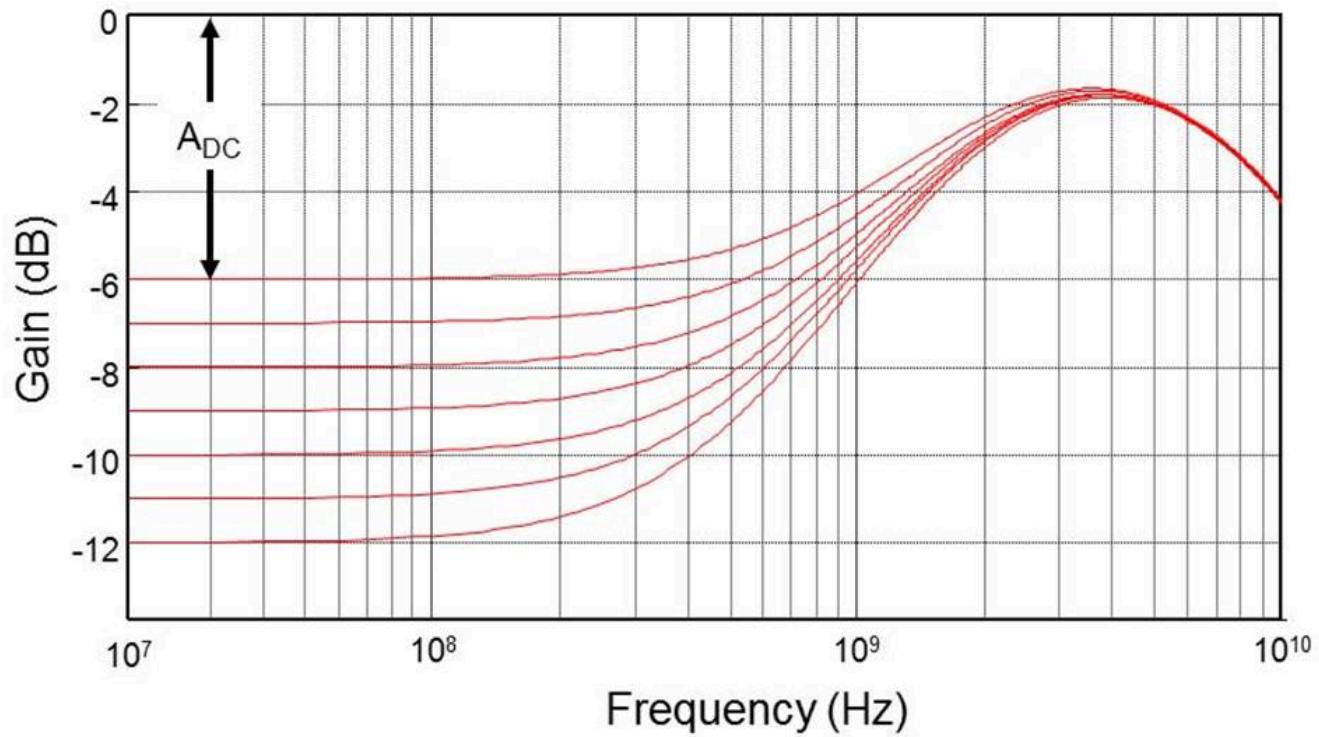


Figure 8-37 Loss Curves for 8.0 GT/s Behavioral CTLE §

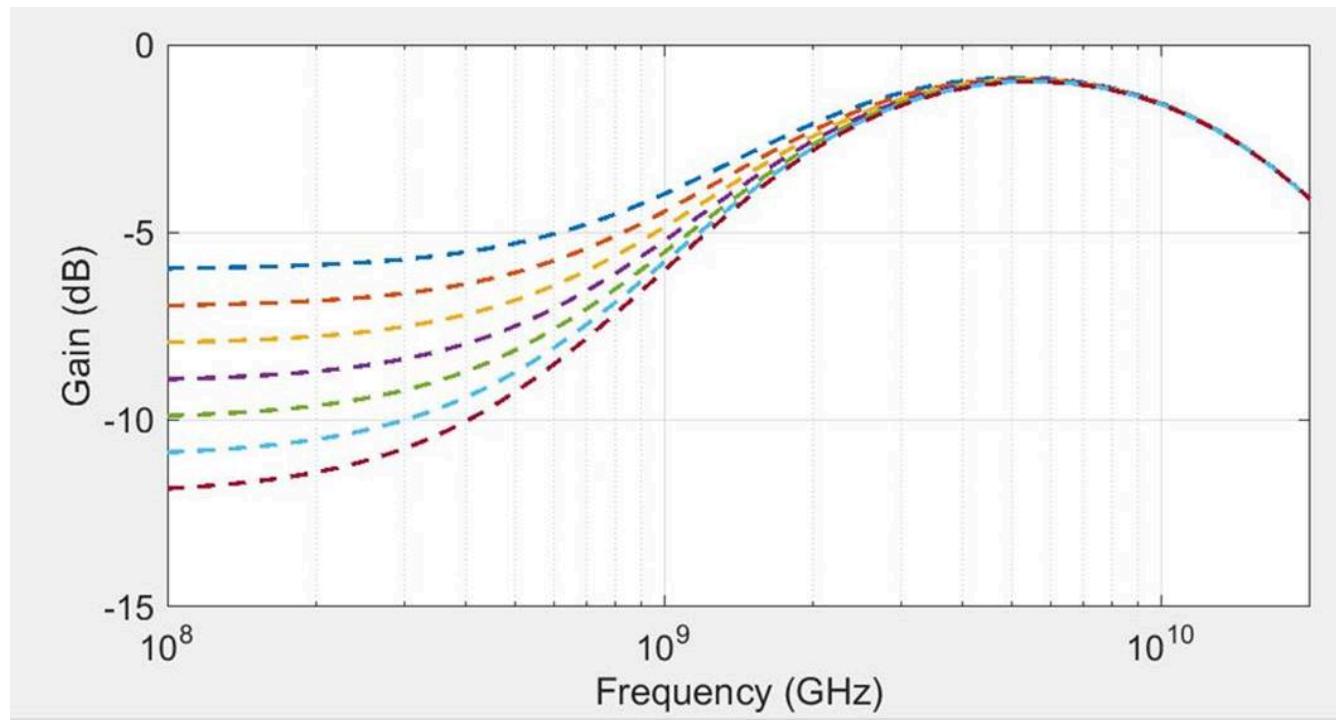


Figure 8-38 Loss Curves for 16.0 GT/s Behavioral CTLE §

A Receiver operating at 16.0 GT/s utilizes a similar set of CTLE curves with different pole locations. The difference is that $\omega_{p1} = \text{pole 1} = 2\pi * 2 \text{ GHz}$ and $\omega_{p2} = \text{pole2} = 2\pi * 16.0 \text{ GHz}$. The range for A_{DC} remains the same as that for 8.0 GT/s.

8.4.1.9 Behavioral CTLE (32.0 and 64.0 GT/s) §

32.0 GT/s behavioral Rx equalization defines a 2nd order CTLE with fixed poles, and an adjustable DC gain (A_{DC}) specified according to the family of curves shown in § Figure 8-39. The A_{DC} is adjustable over a range of -5 to -15 dB in steps of 1.0 dB.

$$H(s) = \frac{\omega_{P1} \times \omega_{P3} \times \omega_{P4}}{\omega_{Z1}} \times \frac{(s + \omega_{Z1})(s + \omega_{P2} \times A_{DC})}{(s + \omega_{P1})(s + \omega_{P2})(s + \omega_{P3})(s + \omega_{P4})}$$

$$\omega_x = 2\pi \times F_x$$

$$F_{P1} = 1.65 \times F_{Z1}$$

$$F_{P2} = 9.5 \text{ GHz}$$

$$F_{P3} = 28 \text{ GHz}$$

$$F_{P4} = 28 \text{ GHz}$$

$$F_{Z1} = 450 \text{ MHz}$$

$$F_{Z2} = \text{mag(DC gain)} \times F_{P2}$$

Equation 8-26 Behavioral CTLE at 32.0 GT/s §

§ Figure 8-39 illustrates the gain vs. frequency behavior of the CTLE as A_{DC} is varied over its minimum to maximum range in 1.0 dB steps. Note that the maximum frequency of the CTLE curves is 200 GHz which ensures accuracy in the time-domain post-processing simulation tools.

Base 6.4 vs Base 6.3

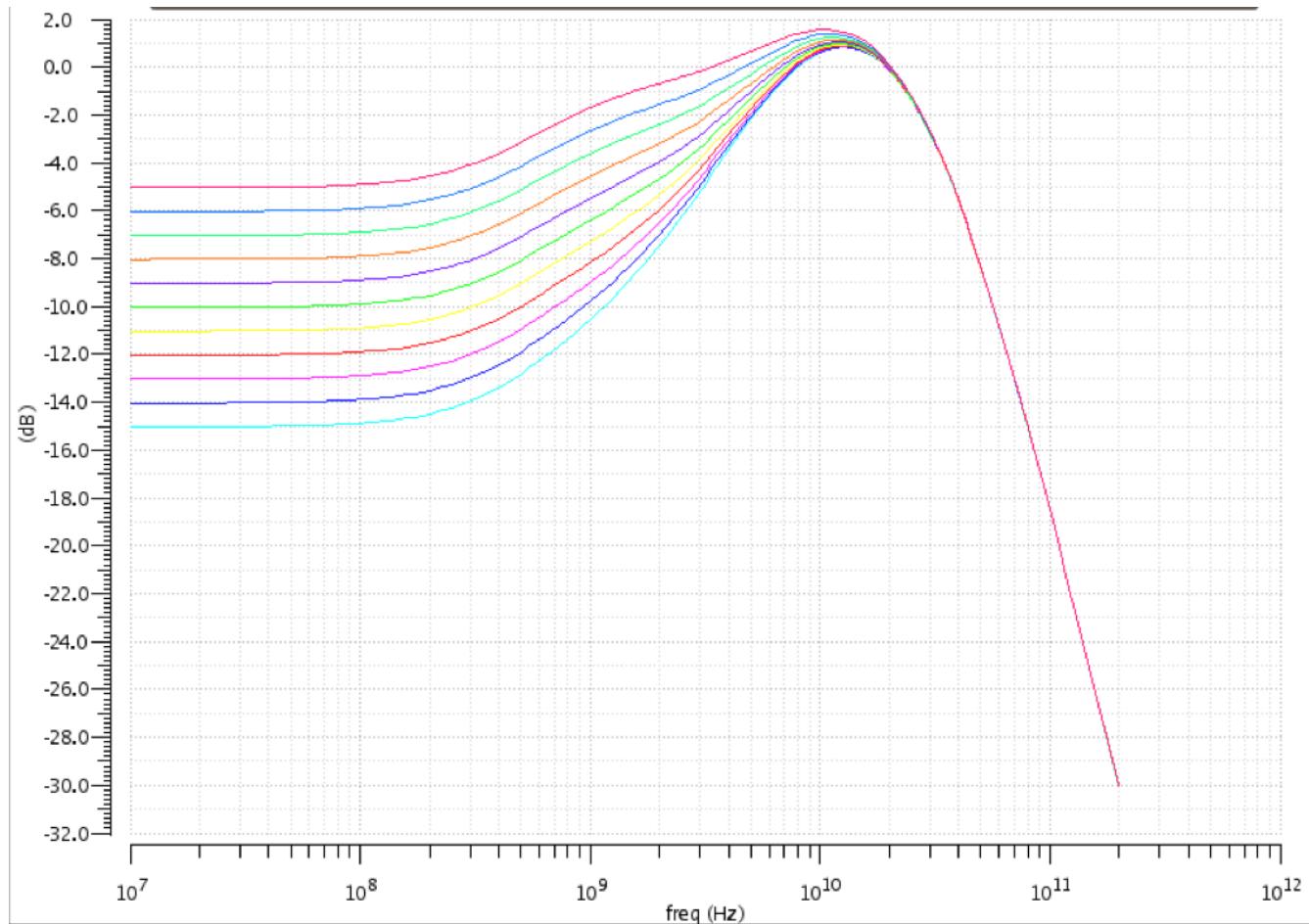


Figure 8-39 Loss Curves for 32.0 GT/s Behavioral CTLE §

64.0 GT/s behavioral Rx equalization defines a CTLE with six poles and three zeros, and an adjustable DC gain (ADC) specified according to the family of curves shown in § Figure 8-40 . The ADC is adjustable over a range of -5 to -15 dB in steps of 1.0 dB. The maximum frequency of the CTLE curves is 250 GHz.

$$H(s) = \frac{\omega_{p1} \times \omega_{p3} \times \omega_{p4} \times \omega_{p5} \times \omega_{p6}}{\omega_{z1} \times \omega_{z3}} \times \frac{(s + \omega_{z1}) \times (s + \omega_{p2} \times A_{DC}) \times (s + \omega_{z3})}{(s + \omega_{p1}) \times (s + \omega_{p2}) \times (s + \omega_{p3}) \times (s + \omega_{p4}) \times (s + \omega_{p5}) \times (s + \omega_{p6})}$$

$$\omega_x = 2\pi \times F_x$$

$$F_{P1} = 1.30 \times F_{z1}$$

$$F_{P2} = 7.7 \text{ GHz}$$

$$F_{P3} = 22.0 \text{ GHz}$$

$$F_{P4} = 28.0 \text{ GHz}$$

$$F_{P5} = 32.0 \text{ GHz}$$

$$F_{P6} = 32.0 \text{ GHz}$$

$$F_{Z1} = 250 \text{ MHz}$$

$$F_{Z2} = \text{mag(DC gain)} \times F_{P2}$$

$$F_{Z3} = 7.7 \text{ GHz}$$

Equation 8-27 Behavioral CTLE at 64.0 GT/s §

Base 6.4 vs Base 6.3

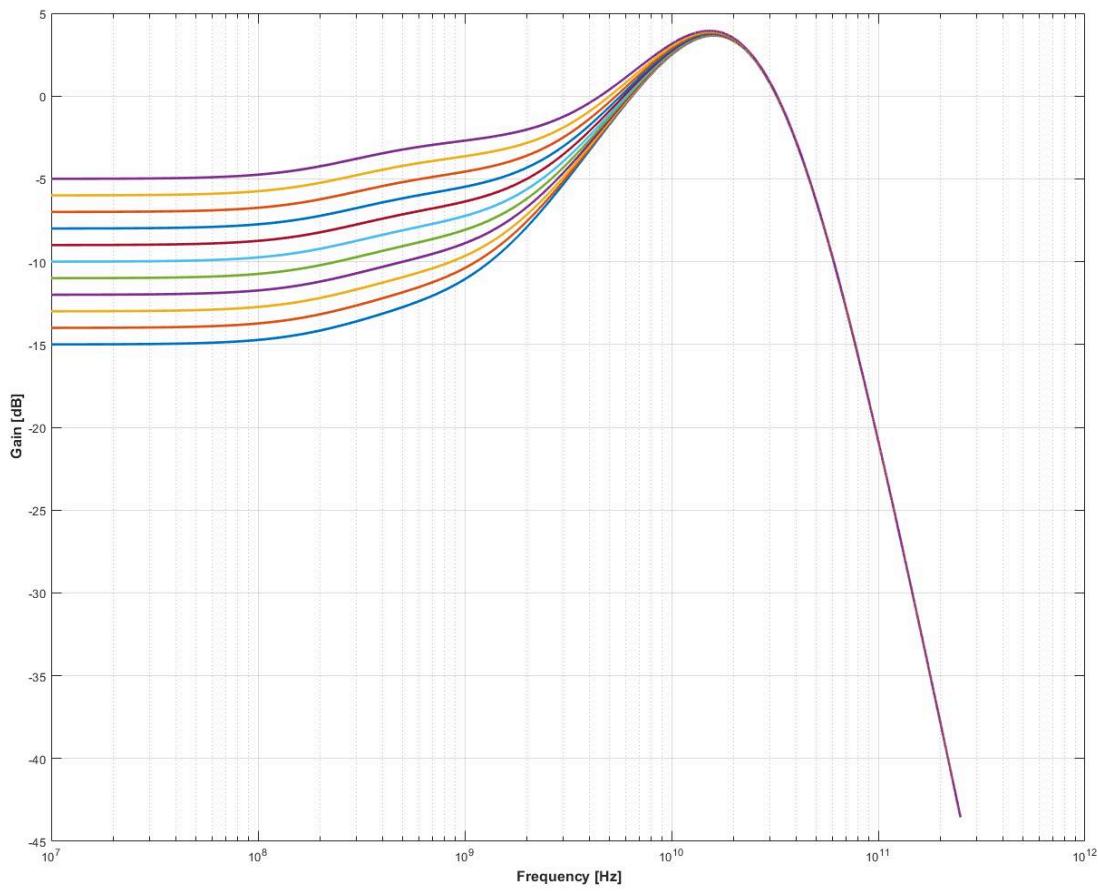


Figure 8-40 Loss Curves for 64.0 GT/s Behavioral CTLE §

8.4.1.10 Behavioral DFE (8.0, 16.0, 32.0, and 64.0 GT/s Only) §

At 8.0 GT/s the combination of a 1st order CTLE and a one-tap DFE algorithm is required for calibrating the stressed eye when employing the max length calibration channel. The DFE may be represented by the following equation and flow diagram. For 8.0 GT/s and 16.0 GT/s the limits on d_1 are ± 30 mV. For 32.0 GT/s the limit on d_1 is defined as a ratio of the tap magnitude (h_1) to the cursor strength (h_0). The h_1/h_0 ratio must be less than or equal to 0.8. Note that the h_1/h_0 limit of 0.8 is only to bound the behavior of the reference receiver and does not indicate that real implementations will be safe from error bursts due to large h_1/h_0 causing undetected data errors if the h_1/h_0 ratio is below 0.8. Implementers must do their own analysis for their specific designs on the largest safe h_1/h_0 ratio. Note that an optional precoding mechanism is provided at 32.0 GT/s that receivers can optionally enable to reduce the risk of DFE related error bursts in high transition data patterns causing silent data corruption. For 16.0 GT/s the limits on d_2 are ± 20 mV. For 32.0 GT/s the limits on d_2 and d_3 are ± 20 mV.

Base 6.4 vs Base 6.3

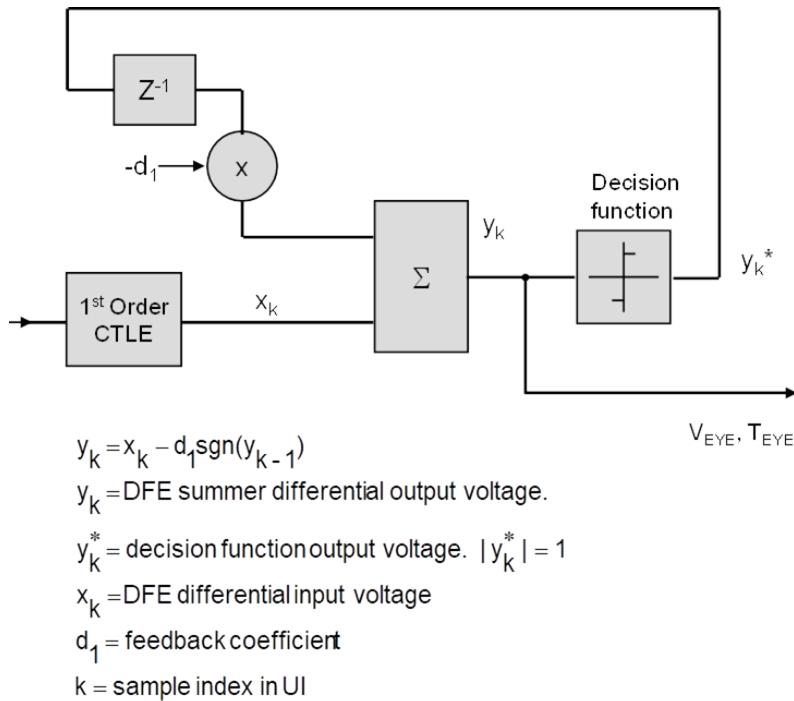


Figure 8-41 Variables Definition and Diagram for 1-tap DFE §

16.0 GT/s Receiver tolerancing utilizes a CTLE and a 2-tap behavioral DFE as illustrated below. Other than the inclusion of the second tap, it is identical to the 1-tap DFE shown above. The 32.0 GT/s Receiver tolerancing utilizes a CTLE and a 3-tap behavior DFE.

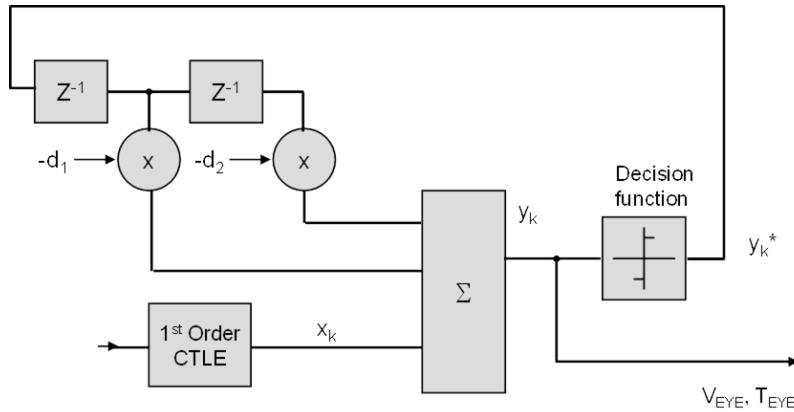


Figure 8-42 Diagram for 2-tap DFE §

For 64.0 GT/s, the feedback signal y_k^* can take values of -1, -1/3, +1/3 and +1. In this case, the limit on d_1 is defined as a ratio of the tap magnitude ($|d_1|$) to the cursor magnitude at the input of the DFE (h_0). To constrain DFE burst errors, the $|d_1/h_0|$ ratio must be less than 0.55 and the weighted-sum of the tap magnitudes defined as $(|d_1| + |d_2| + 0.85*|d_3| + 0.60*|d_4| + 0.25*|d_5| + 0.10*|d_6| + 0.05*|d_7| + 0.05*|d_8| + 0.05*|d_9| + 0.05*|d_{10}| + 0.05*|d_{11}| + 0.05*|d_{12}| + 0.05*|d_{13}| + 0.05*|d_{14}| + 0.05*|d_{15}| + 0.05*|d_{16}|)/h_0$ must be less than 0.85. The limits on DFE tap magnitudes are only to bound the behavior of the reference receiver and do not indicate that real implementations will be safe from error bursts by

ensuring the DFE tap magnitude limits specified for the reference receiver. Implementers must do their own analysis for their specific designs on the largest safe DFE tap magnitudes.

8.4.2 Stressed Eye Test §

Rx testing at 16.0, 32.0, and 64.0 GT/s requires only a single stressed voltage/stressed jitter test per data rate.

When testing a Receiver, it is required to have other PCI Express Lanes on the DUT sending or receiving data. Similarly, if the device supports other I/O, it should also be sending or receiving on these interfaces. The goal is to have the Rx test environment replicate the noise environment found in a real system as closely as possible.

8.4.2.1 Procedure for Calibrating a Stressed EH/EW Eye §

The goal of calibrating a stressed voltage/jitter eye is to present the Receiver under test with simultaneously worst case margins whose distortion characteristics are like an eye produced by a real channel. Much of the distortion consists of the ISI produced by the calibration channel. Incremental changes of R_j and differential voltage are allowed to adjust the EW and EH, respectively at 8.0 GT/s. Incremental changes of S_j , $V_{RX-DIFF-INT}$, and differential voltage swing from nominal values may be used to adjust the EW and EH at 16.0, 32.0, and 64.0 GT/s. Refer to § Table 8-11 for initial values of various stress parameters for all data rates.

The reference point where EH/EW is defined corresponds to input to the Receiver latch at 8.0, 16.0, 32.0, and 64.0 GT/s. Since this point is not physically accessible it is necessary to construct its equivalent by means of a post-processing procedure. A two million unit interval data record of compliance pattern or a step that has been averaged 1024 times at TP2 is first post processed to mathematically include the additional signal distortion caused by the behavioral Receiver package. If a compliance pattern waveform is used then all stresses except $V_{RX-CM-INT}$ are turned on if a step is used then all stresses are turned off. Then the resulting signal is recovered by means of Rx equalization, and a behavioral CDR function, resulting in an equivalent eye. The requirements for the waveform post processing tool used for the EH/EW calibration are described further in § Section 8.4.2.1.1. If the receiver calibration eye margin simulation tool uses a step response, the R_j , S_j , and $V_{RX-DIFF-INT}$ are input parameters to the simulation tool. If the receiver calibration eye margin simulation tool uses compliance pattern waveform, the R_j , S_j , and $V_{RX-DIFF-INT}$ are inputs to the waveform. In either case, the stress parameters must be calibrated.

As the calibration procedure of the signal generator output contains steps where the generator is connected directly to measurement instrumentation, the transition time of the output waveform can be very fast. Therefore, it is important that the bandwidth of instrumentation used to calibrate the generator be matched appropriately to the edge rate of the generator output. This specification requires the use of a generator for 16.0 GT/s testing whose outputs have a rise time of 14 ps-19 ps (20% / 80%) which also requires a minimum oscilloscope bandwidth of 25 GHz. This oscilloscope bandwidth is also the minimum required bandwidth for transmitter measurements at 16.0 GT/s. For 32.0 and 64.0 GT/s testing the specification requires the use of a generator whose outputs have a rise time of 7.5 - 15.0 ps (20%/80% measured with P4) which requires a minimum oscilloscope bandwidth of 50 GHz. This oscilloscope bandwidth is also the minimum required for transmitter measurements at 32.0 and 64.0 GT/s. A minimum oscilloscope sampling rate that captures at least 4 samples per unit interval is required for all data rates.

For the eye calibration process, the Tx equalization is fixed to the preset that gives the optimal eye area with the post processing tool being used for calibration. Once the testing procedure is under way the Tx preset may be adjusted to yield the best eye margins with the DUT. During EH/EW calibration S_j is initially set to 100 MHz with a nominal amplitude of 0.1 UI for 8.0, 16.0, and 32.0 GT/s, with a nominal amplitude of 0.05 UI for 64.0 GT/s. The 100 MHz S_j amplitude will be swept during the stressed eye calibration. Tx EQ and differential voltage swing calibration are done at TP3 as shown in § Figure 8-28. The coaxial cable from TP1 to TP3 is considered part of the generator and not included in the channel insertion loss measurements for 32.0 GT/s and 64.0 GT/s stressed eye calibration, but the coaxial cable is included in the total channel insertion loss measurement at 16.0 GT/s to keep consistency with the 16.0 GT/s measurement methodology adopted in PCIe 4.0. For calibration at 16.0 GT/s the following process is used to calibrate the eye:

1. Calibrate the stress values to the nominal values in § Table 8-11 .
2. Select an initial test channel length that gives a loss at TP2P at 8 GHz of $27 \text{ dB} \pm 0.5 \text{ dB}$.
3. Measure the eye diagram for each TX EQ preset using the nominal TX Eq for the preset $\pm 0.1 \text{ dB}$ and select the TX EQ preset that gives the largest eye area.

For all EH, EW and eye area measurements performed in receiver calibration the A_{DC} in the reference receiver CTLE is varied over its minimum to maximum range in 0.25 dB steps. This is done to improve repeatability and accuracy in automated Rx calibration software and is only done for stressed eye calibration (not for channel compliance, etc.)

4. Increase the calibration channel loss to the next available length/loss and measure the new eye diagram at the selected preset. Continue to increase the length/loss until either the height or width have fallen below the targets in § Table 8-11 then the previous calibration channel length/loss is selected. If neither the height or width have fallen below the targets and the TP3 (§ Figure 8-28) to TP2P loss at 8 GHz has reached 30.0 dB then advance to the next step.
5. For the selected calibration channel length/loss, measure the eye diagram for each TX EQ preset and select the preset that gives the largest eye area. Note that this may be a different preset than step 3 due to the length/loss change.
6. Adjust S_j , $V_{RX-DIFF-INT}$, and Voltage Swing to make final adjustments to the eye by sweeping them through the following ranges:
 - a. S_j 5 to 10 ps PP.
 - b. $V_{RX-DIFF-INT}$ 10 to 25 mV at TP2.
 - c. Differential Voltage Swing 720 to 800 mV PP at TP1.
7. If the final S_j value is less than 0.1 UI then the R_j level is reduced so the eye width meets the target eye width with 0.1 UI of 100 MHz S_j .
8. If there are multiple combinations of S_j , $V_{RX-DIFF-INT}$, and Voltage Swing that give valid solutions first pick the combination that is closest to the target eye width (18.75 ps). If there are multiple S_j , $V_{RX-DIFF-INT}$, and Voltage Swing combinations that are equally close to the target eye width then pick the one with S_j closest to nominal. The selected values must give a mean eye height and width (over at least 5 measurements exact number of measurements needed for stable values will depend on lab set-up and tools) within the following ranges at BER E-12:
 - a. Eye height 15 mV $\pm 1.5 \text{ mV}$
 - b. Eye width 18.75 ps $\pm 0.5 \text{ ps}$

For calibration at 32.0 GT/s the following process is used to calibrate the eye:

1. Measure the eye diagram for each TX EQ preset using the nominal TX Eq for the preset $\pm 0.3 \text{ dB}$ and select the TX EQ preset that gives the largest eye area.

For all EH, EW and eye area measurements performed in receiver calibration the A_{DC} in the reference receiver CTLE is varied over its minimum to maximum range in 1.0 dB steps. Measure the eye diagram varying the following parameters with the maximum indicated step size for each variable parameter:

- a. Calibrate the stress values to the initial values in § Table 8-11 . The R_j stress is kept constant.
- b. Channel loss from TP3 to TP2P varied from 34.0 to 37.0 dB with a maximum 0.5 dB step size.

Note that if your actual channel loss comes out to slightly above 37 or slightly below 34 that these cases are excluded (loss must be between 34.0 and 37.0 dB)

- c. S_j varied from 1 to 5 ps PP with a maximum 0.25 ps step size with S_j measured at TP3

- d. $V_{RX-DIFF-INT}$ 5 to 30 mV at TP2 with a maximum 2.5 mV step size
- 2. If the final S_j value is less than 0.1 UI then the R_j level is reduced so the eye width meets the target eye width with 0.1 UI of 100 MHz S_j .
- 3. If there are multiple combinations of S_j , $V_{RX-DIFF-INT}$, and channel loss that give valid solutions first pick the combination with the highest channel loss. If there are multiple combinations that work with the highest channel loss, then select the combination that is closest to the target eye height (15.0 mV). The selected values must give a mean eye height and width (over at least 5 measurements exact number of measurements needed for stable values will depend on lab set-up and tools) within the following ranges at BER E-12. A specific method for finding the combination of stress values that meet these criteria is outside the scope of this specification. If and only if no stress combinations can be found, then the voltage swing may also be varied from 720 to 800 mV:

Note that because the first tiebreaker is highest loss - most approaches will start with the highest allowed channel loss.

- a. Eye height 15 mV +/- 1.5 mV
- b. Eye width 9.375 ps +/- 0.5 ps

For calibration at 64.0 GT/s the following process is used to calibrate the eye:

- 1. Measure the PAM4 eye diagrams for each TX EQ preset of Q0-Q9 using the nominal TX Eq for the preset +/- 0.3 dB and select the TX EQ preset that gives the largest top eye area.

For all EH, EW and eye area measurements performed in receiver calibration the ADC in the reference receiver CTLE is varied over its minimum to maximum range in 1.0 dB steps. Measure the eye diagrams varying the following parameters with the maximum indicated step size for each variable parameter:

- a. Calibrate the stress values to the initial values in § Table 8-11 . The R_j stress is kept constant.
- b. Channel loss from TP3 to TP2P varied from 30.0 to 33.0 dB with a maximum 0.5 dB step size.
- c. S_j varied from 1 to 3 ps PP with a maximum 0.25 ps step size with S_j measured at TP3
- d. $V_{RX-DIFF-INT}$ 5 to 25 mV at TP2 with a maximum 2.0 mV step size
- 2. If the final S_j value is less than 0.05 UI then the R_j level is reduced so the eye width meets the target eye width with 0.05 UI of 100 MHz S_j .
- 3. If there are multiple combinations of S_j , VRX-DIFF-INT, and channel loss that give valid solutions first pick the combination with the highest channel loss. If there are multiple combinations that work with the highest channel loss, then select the combination that is closest to the target top eye height (6.0 mV). The selected values must give a mean eye height and width (over at least 5 measurements exact number of measurements needed for stable values will depend on lab set-up and tools) within the following ranges at BER of 10-6. A specific method for finding the combination of stress values that meet these criteria is outside the scope of this specification. If and only if no stress combinations can be found, then the voltage swing between PAM4 voltage level 0 and level 3 may also be varied from 720 to 800 mV:

Note that because the first tiebreaker is highest loss - most approaches will start with the highest allowed channel loss.

- a. Top eye height 6 mV +/- 0.5 mV
- b. Top eye width 3.125 ps +/- 0.3 ps

For calibration at 64.0 GT/s, if the Differential Noise and/or Common Mode Noise are generated using an external source, any broadband differential noise introduced by the noise source should be characterized and added to the step response based calibration procedure through inclusion of SNDR in the modeling of the BERT transmitter.

Based upon the data rate at which the Rx is being tested, R_j or S_j and differential interference sources are adjusted to fall within the V_{RX-ST} and T_{RX-ST} limits. The EH and EW ranges are designed to account for post processing or measurement errors. § Figure 8-43 shows the process for calibrating the stressed jitter eye at 8.0 GT/s.

Base 6.4 vs Base 6.3

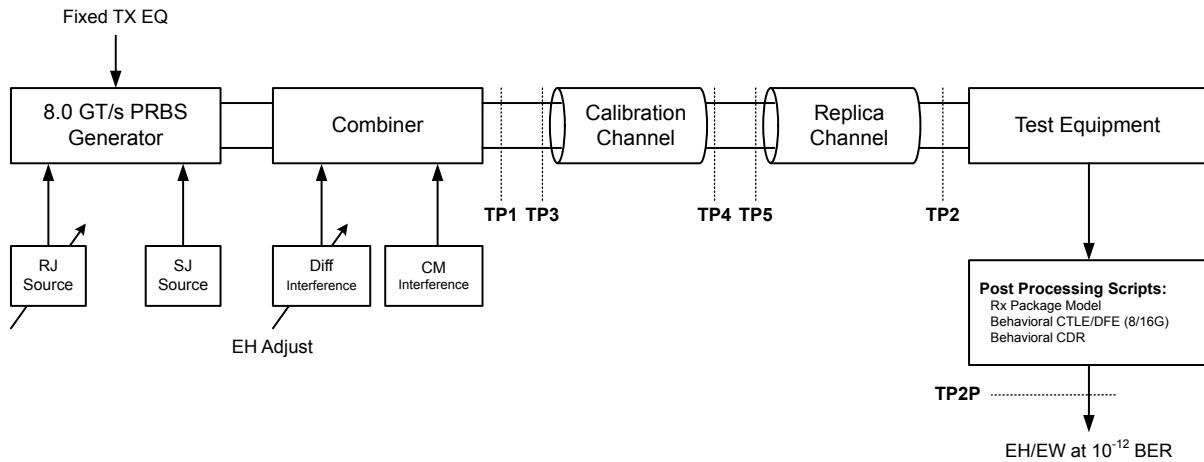


Figure 8-43 Layout for Calibrating the Stressed Jitter Eye at 8.0 GT/s §

§ Figure 8-44 shows the process for calibrating the stressed jitter eye common for 16.0, 32.0, and 64.0 GT/s data rates. The PRBS Generator provides the required data rate and the eye is calibrated to the data rate specific target EH and EW at the corresponding BER.

Base 6.4 vs Base 6.3

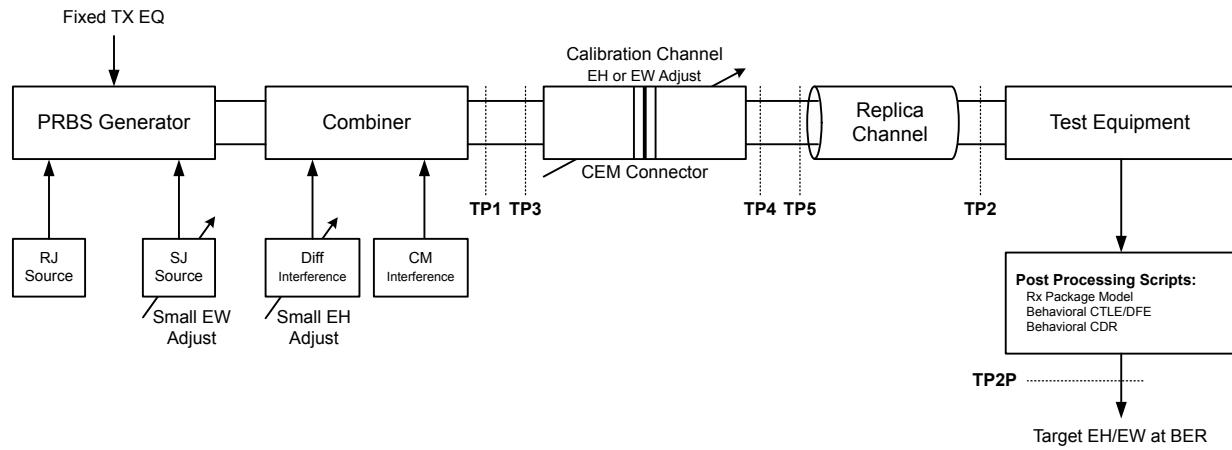


Figure 8-44 Layout for Calibrating the Stressed Jitter Eye at 16.0, 32.0, and 64.0 GT/s §

Table 8-11 Stressed Jitter Eye Parameters §

| Symbol | Parameter | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s | Units | Details |
|-------------------|--------------------------------------|-------------|----------------------------|-------------------------|-----------|-----------|-----------|--------|--|
| $V_{RX-LAUNCH}$ | Generator launch voltage | 800 to 1200 | 800 to 1200 | 800 to 1200 | 800 | 800 | 800 | mV PP | Note 1 |
| T_{RX-UI} | Unit Interval | 400 | 200 | 125 | 62.5 | 31.25 | 31.25 | ps | |
| T_{RX-ST} | Target Eye width (Top Eye for PAM4) | 0.4 | 0.32 | 0.30 | 0.30 | 0.30 | 0.10 | UI | Note 3, 4, 8, 10 |
| V_{RX-ST} | Target Eye height (Top Eye for PAM4) | 175 | 100 | 25 | 15 | 15 | 6 | mV PP | Note 2, 4, 8, 9 |
| $T_{RX-ST-SJ}$ | Swept Sj | N/A | 75 ps (max) See Note 11 | See § Section 8.4.2.2.1 | | | | | ps |
| $T_{RX-ST-RJ}$ | Random Jitter | N/A | 3.4 | (max) 3.0 | 1.0 | 0.5 | 0.25 | ps RMS | Note 6, 7 |
| $V_{RX-DIFF-INT}$ | Differential noise | N/A | N/A | 14 | 14 | 20 | 15 | mV PP | Note 7, 12 Adjust to set EH. Frequency = 2.1 GHz |
| $V_{RX-CM-INT}$ | Common mode noise | 150 | 150 | 150 | 150 | 150 | 75 | mV PP | Note 8 |
| $V_{SSC-RES}$ | SSC Residual | N/A | 75 | N/A | 500 | N/A | N/A | ps | Note 11, 13 |

Notes:

| Symbol | Parameter | 2.5 GT/s | 5.0 GT/s | 8.0 GT/s | 16.0 GT/s | 32.0 GT/s | 64.0 GT/s | Units | Details |
|---|-----------|----------|----------|----------|-----------|-----------|-----------|-------|---------|
| <p>1. $V_{RX-LAUNCH}$ may be adjusted to meet V_{RX-ST} as long as the outside eye voltage at TP2 does not exceed 1300 mVPP for calibration at 2.5, 5.0, and 8.0 GT/s. $V_{RX-LAUNCH}$ is adjusted from 800 to 720 mV for 16.0, 32.0, and 64.0 GT/s calibration with 800 mV as the nominal value.</p> <p>2. Voltages shown for 2.5 GT/s and 5.0 GT/s are at the Rx pins.</p> <p>3. Eye widths shown for 2.5 GT/s and 5.0 GT/s are at the Rx pins.</p> <p>4. V_{RX-ST} and T_{RX-ST} are referenced to TP2P for 8.0, 16.0, 32.0, and 64.0 GT/s and TP2 for 2.5 and 5.0 GT/s. For 8.0, 16.0, 32.0, and 64.0 GT/s behavioral equalization are applied to the data at TP2. At 64.0 GT/s, V_{RX-ST} and T_{RX-ST} correspond to top eye height and eye width at 10^{-6} BER.</p> <p>5. $T_{RX-ST-SJ}$ may be measured at either TP1 or TP2. Only 8.0, 16.0, 32.0 and 64.0 GT/s receivers are tested with SJ mask.</p> <p>6. $T_{RX-ST-RJ}$ may be adjusted to meet the target value for T_{RX-ST} at 8.0 GT/s. Rj is measured at TP1 to prevent data-channel interaction from adversely affecting the accuracy of the Rj calibration. Rj is applied over the following range: The low frequency limit may be between 1.5 and 10 MHz, and the upper limit is 1.0 GHz.</p> <p>7. Both $T_{RX-ST-RJ}$ and $V_{RX-DIFF-INT}$ are limited to prevent the stressed eye from containing excessive amounts of jitter or noise distortion that are unrepresentative of a real channel. Too many of these distortion components produces a signal that cannot be equalized by an actual Receiver.</p> <p>8. Defined as a single tone at 120 MHz. Measurement made at TP2 without post-processing. Common mode is turned off during T_{RX-ST} and V_{RX-ST} calibration and then turned on for the stressed eye jitter test.</p> <p>9. For 2.5 GT/s and 5.0 GT/s Rx calibration variable channel loss is used to achieve the target eye height.</p> <p>10. For 2.5 GT/s Rx calibration 100 MHz SJ is used to achieve the target eye width.</p> <p>11. For 33 kHz SSC residual for common clock architecture testing only when testing at 5 GT/s.</p> <p>12. Frequency for $V_{RX-DIFF-INT}$ is chosen to be slightly above the first pole of the reference CTLE.</p> <p>13. Applied for CC testing only as a triangular phase modulation with a frequency between 30 kHz to 33 kHz when testing at 16.0 GT/s with no 32.0 GT/s and no 64.0 GT/s support and when the SJ mask of § Figure 8-52 and a first order CDR transfer function are used.</p> | | | | | | | | | |

8.4.2.1.1 Post Processing Tool Requirements §

A waveform post processing tool or a channel compliance methodology tool may be used for Rx stressed eye calibration at 16.0, 32.0, or 64.0 GT/s. If a waveform post processing tool is used to calibrate the EH/EW for the RX stressed eye testing, the tool must be consistent with the channel compliance methodology tool based on a consistency test defined as follows:

- The test channel is the long Rx calibration channel with the Root reference package applied in post-processing to give a total loss of 28.0 dB at 8 GHz (16.0 GT/s), 36.0 dB at 16 GHz (32.0 GT/s), or 32.0 dB at 16 GHz (64.0 GT/s). This means that the physical channel loss is 23.0 dB (16.0 GT/s), 27.0 dB (32.0 GT/s), or 24.0 dB (64.0 GT/s).
- All measurements are done at TP2.
- A step pattern with 512 ones and zeros is captured through the test channel by averaging 1024 times on a real time oscilloscope. The step is saved with an x-axis resolution of 1 ps or less to be used as the transmit waveform for the channel compliance methodology. The step pattern is captured for each preset using the nominal Tx EQ for each preset.
- The channel compliance methodology is run with no Tx EQ applied in simulation using the nominal stress values for Rx stressed eye calibration for each of the captured steps. The Tx EQ preset that produced the largest eye area is selected for exact eye height and width calibration.

- The channel compliance methodology is used with the selected Tx EQ preset to produce an EH/EW of
 - 15 mV and 0.3 UI @ 10^{-12} BER (16.0 GT/s),
 - 15 mV and 0.3 UI @ 10^{-12} BER (32.0 GT/s), or
 - top eye height of 6.0 mV and top eye width of 0.1 UI @ 10^{-6} BER (64.0 GT/s)
- by adjusting the S_j , $V_{RX\text{-}DIFF\text{-}INT}$ and voltage swing at the Transmitter output.
- A pattern generator is calibrated to have the same jitter stress levels and Tx Swing as those used in the channel compliance simulations that produced the target eye height and eye width.
 - 2 million unit interval waveforms with compliance pattern are captured at each Tx EQ at the end of the channel. The Tx EQ is calibrated to the nominal values for each preset at the pattern generator output before doing the captures.
 - For the preset that gives the largest eye area with the waveform post processing tool the EH and EW (@ 10^{-12} BER for 8.0, 16.0, and 32.0 GT/s and @ 10^{-6} BER for 64.0 GT/s) & must match the target EH and EW from the channel compliance methodology within +/- 15%.

8.4.2.2 Procedure for Testing Rx DUT §

8.4.2.2.1 S_j Mask §

Once a calibrated EH and EW have been obtained, the cables are moved to connect the Rx DUT to the far end of calibration channel. For the testing of the Rx DUT, the BERT Tx must transmit Modified Compliance Pattern. The Tx equalization may then be optimized with the assumption that the DUT Rx will also optimize its equalization. S_j is set to an initial value of 0.1 UI at 100 MHz and the Receiver CDR must achieve lock. For 64.0 GT/s, S_j is set to an initial value of 0.05 UI at 100 MHz and the Receiver CDR must achieve lock. At 8.0, 16.0, 32.0, and 64.0 GT/s the 100 MHz S_j initial tone is removed and then the appropriate swept S_j profile is tested. At 16.0, 32.0, and 64.0 GT/s an additional S_j tone at 210 MHz is present for all testing. At 16.0 and 32.0 GT/s, the amplitude of this additional tone is equal to the amplitude of the 100 MHz S_j required to achieve the target eye width minus 0.1 UI. If the calibration S_j level was less than 0.1 UI then no additional tone at 210 MHz is used. At 64.0 GT/s, the amplitude of this additional tone is equal to the amplitude of the 100 MHz S_j required to achieve the target eye width minus 0.05 UI. If the calibration S_j level was less than 0.05 UI then no additional tone at 210 MHz is used. Different S_j profiles are used, depending on data rate and whether the Rx under test operates in the CC mode or the IR Refclk mode. See § Table 8-11.

The S_j (pp value) profiles shown in § Figure 8-45 , § Figure 8-46 , § Figure 8-47 , § Figure 8-48 , § Figure 8-49 , § Figure 8-50 , and § Figure 8-51 consist of swept tones at 33 kHz and from 400 kHz to 100 MHz, representing the swept S_j frequency range. For 8.0 and 16.0 GT/s SRIS mode with 5000 ppm SSC, the magnitude of the 33 kHz spur is 25 ns pp. For 32.0 and 64.0 GT/s SRIS mode with 3000 ppm SSC, the magnitude of the 33 kHz spur is 15 ns pp. A Receiver must meet the target BER (10^{-6} for 64.0 GT/s and 10^{-12} for all other data rates) over the entire swept S_j frequency range. It is not necessary to test a Receiver over the entire S_j frequency range, but a sufficient number of frequency points should be tested to guarantee that the Rx does not fail the target BER at some resonance frequency. The 33 kHz frequency point must be tested and treated as another frequency point. The 33 kHz frequency will not be kept on during other frequency measurements. Note that no SSC is applied at the source. Swept S_j is required only for testing Receivers at 8.0, 16.0, 32.0, and 64.0 GT/s. Receiver operation at 2.5 GT/s and 5.0 GT/s is tested using a single 33 kHz S_j tone.

Receivers operating at 8.0 GT/s in the IR mode use the S_j mask profile shown in § Figure 8-45 . The magnitude of the 33 kHz spur is 25 ns pp, or 200 UIpp. The equation of the swept S_j curve is shown on § Figure 8-45 .

Base 6.4 vs Base 6.3

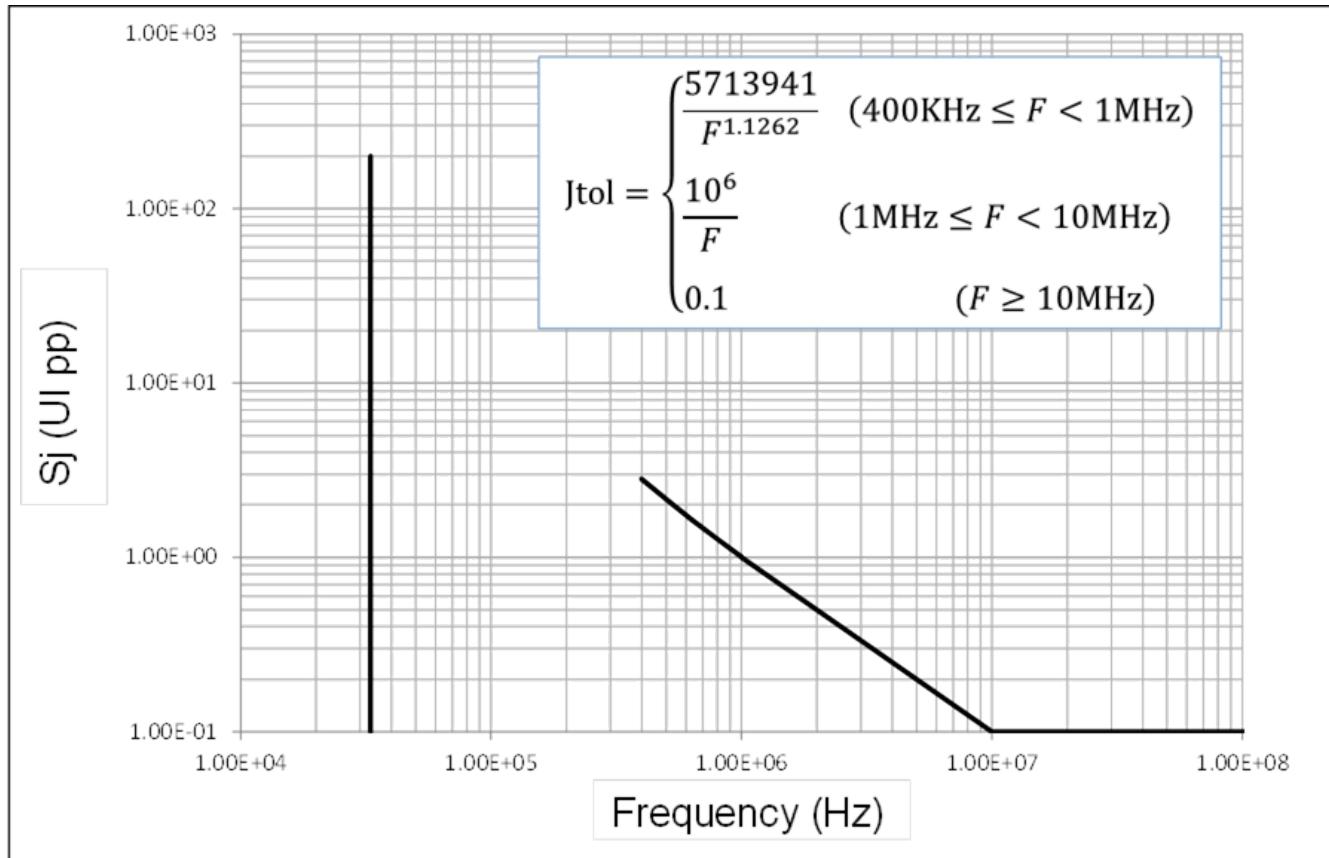


Figure 8-45 *S_j* Mask for Receivers Operating in IR mode at 8.0 GT/s [§](#)

Receivers operating at 16.0 GT/s in the Independent Refclk (IR) mode use the *S_j* mask profile shown in [§ Figure 8-46](#). The magnitude of the 33 kHz spur is 25 ns pp, or 400 UIpp. The equation of the swept *S_j* curve is shown on the [§ Figure 8-46](#).

Base 6.4 vs Base 6.3

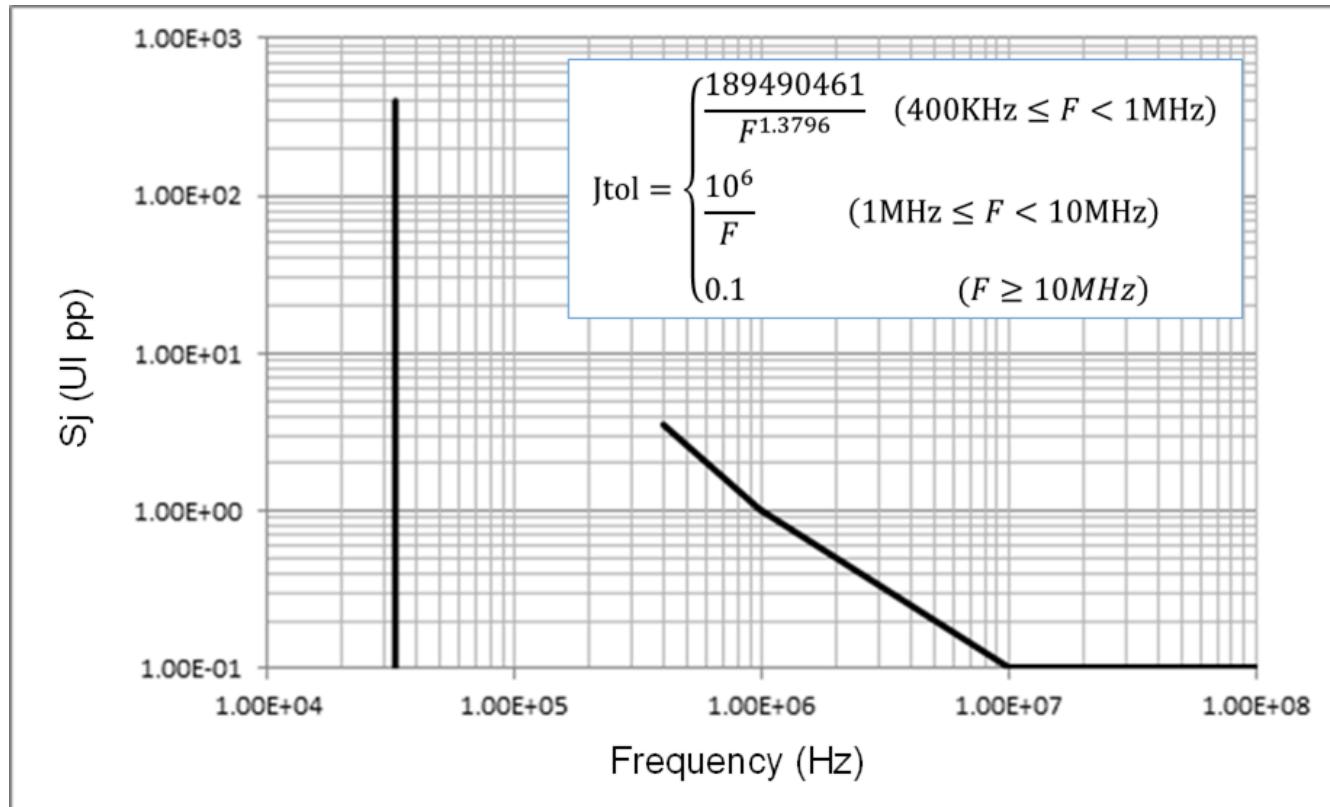


Figure 8-46 S_j Mask for Receivers Operating in SRIS mode at 16.0 GT/s §

Receivers operating at 16.0 GT/s in the Common Clock (CC) Refclk mode use the S_j mask profile shown in § Figure 8-47 . The magnitude of the 33 kHz spur is 1 ns pp, or 16 UIpp. The equation of the swept S_j curve is shown on the § Figure 8-47 . Devices that do not support 32.0 GT/s have the option to use the S_j mask defined in § Figure 8-52 . In this case, a 500 ps-pp triangular SSC modulation has to be applied at the source for both channel calibration and RX compliance, as specified in § Table 8-11 .

Base 6.4 vs Base 6.3

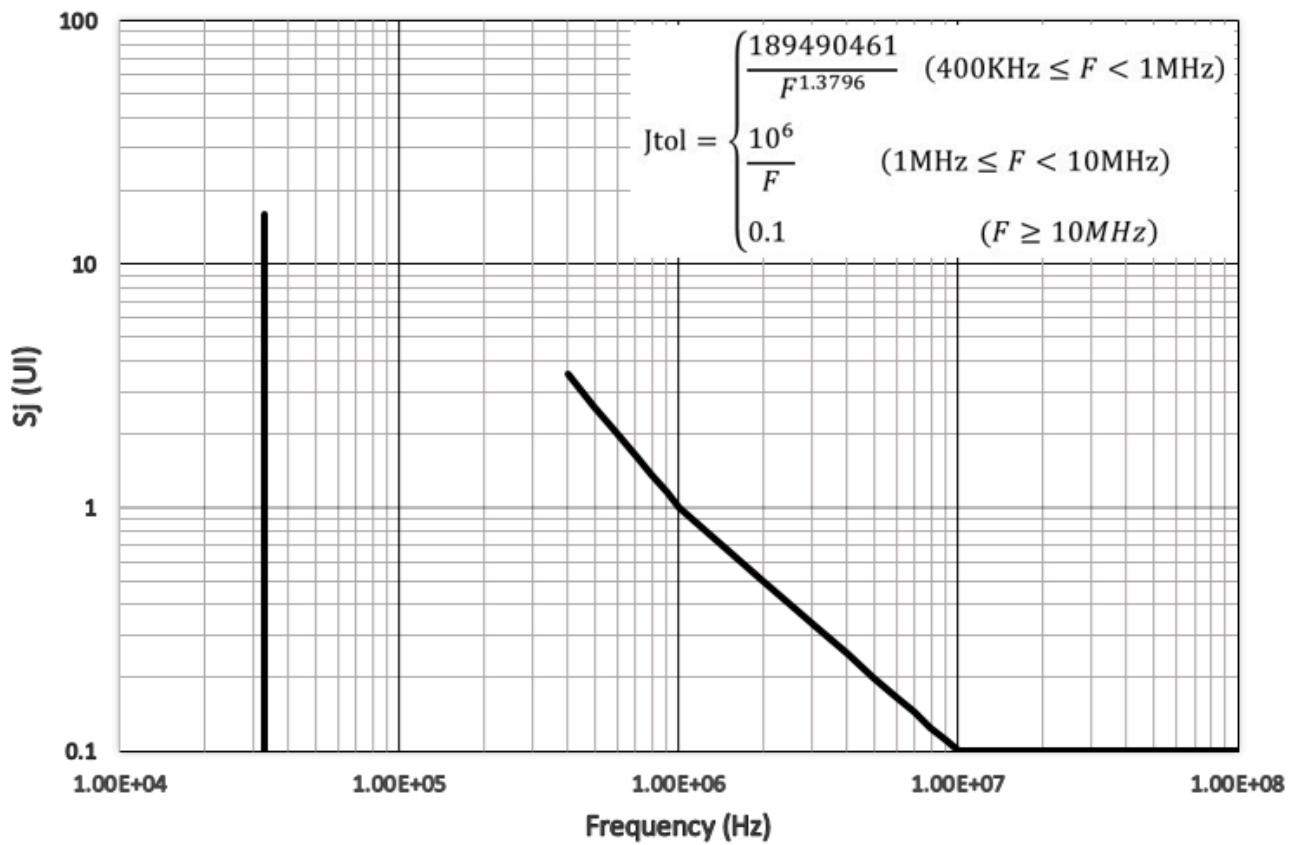


Figure 8-47 S_j Mask for Receivers Operating in CC mode at 16.0 GT/s §

Receivers operating at 32.0 GT/s in the IR mode use the S_j mask profile shown in § Figure 8-48 . The magnitude of the 33 kHz spur is 15 ns pp, or 480 UIpp. The equation of the swept S_j curve is shown on the § Figure 8-48 .

Base 6.4 vs Base 6.3

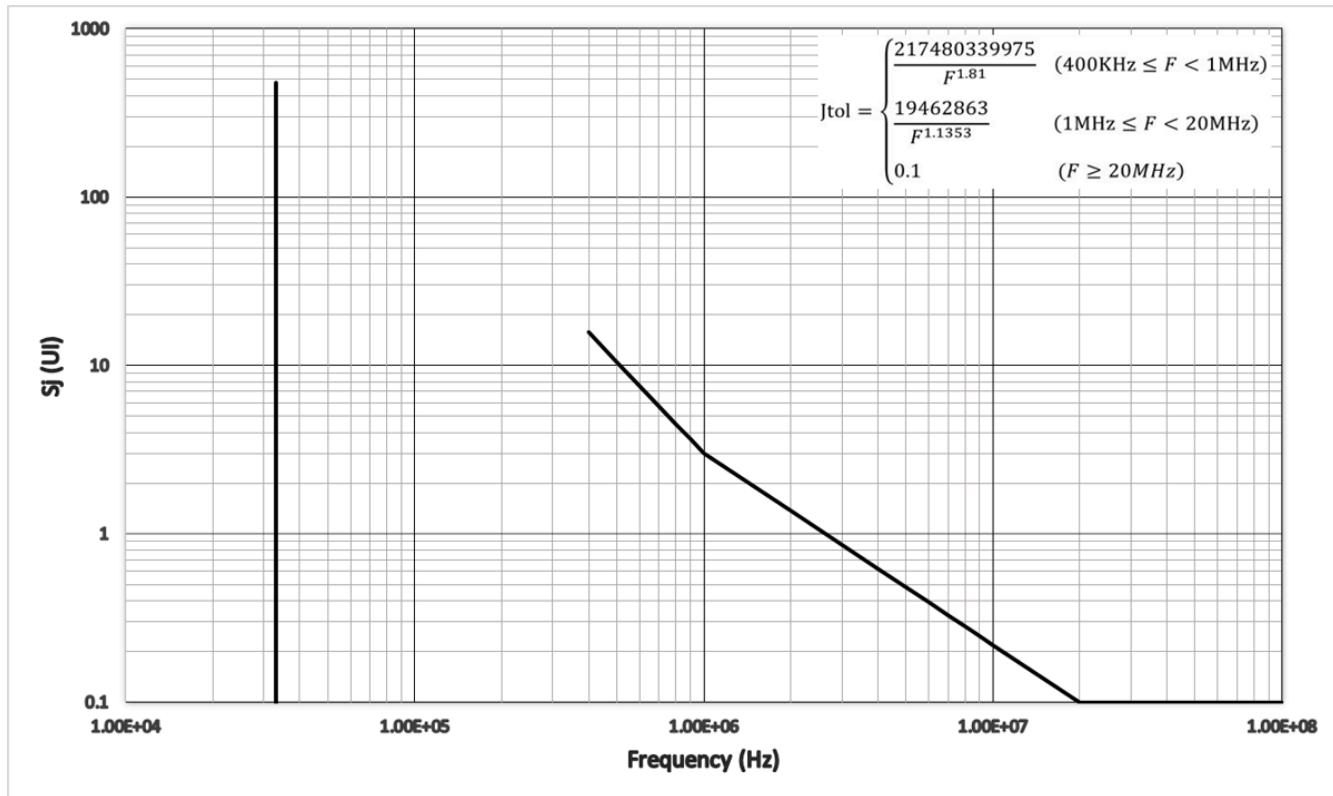


Figure 8-48 Sj Mask for Receivers Operating in SRIS mode at 32.0 GT/s

Receivers operating at 32.0 GT/s in the CC Refclk mode use the Sj mask profile shown in § Figure 8-49 . The magnitude of the 33 kHz spur is 1 ns pp, or 32 UIpp. The equation of the swept Sj curve is shown on the § Figure 8-49 .

Base 6.4 vs Base 6.3

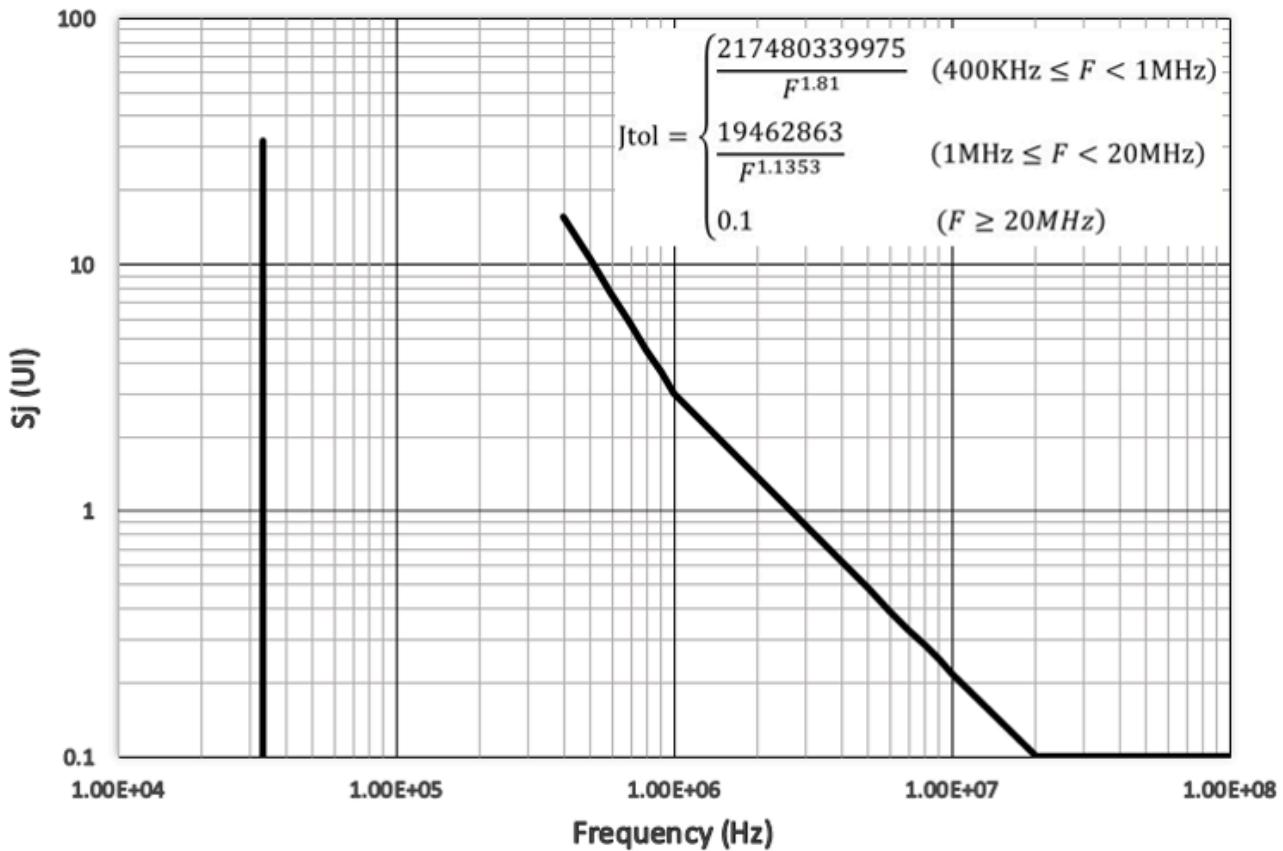


Figure 8-49 S_j Mask for Receivers Operating in CC mode at 32.0 GT/s §

Receivers operating at 64.0 GT/s in the IR mode use the S_j mask profile shown in § Figure 8-50 . The magnitude of the 33 kHz spur is 15 ns pp, or 480 UI pp. The equation of the swept S_j curve is shown on the § Figure 8-50 .

Base 6.4 vs Base 6.3

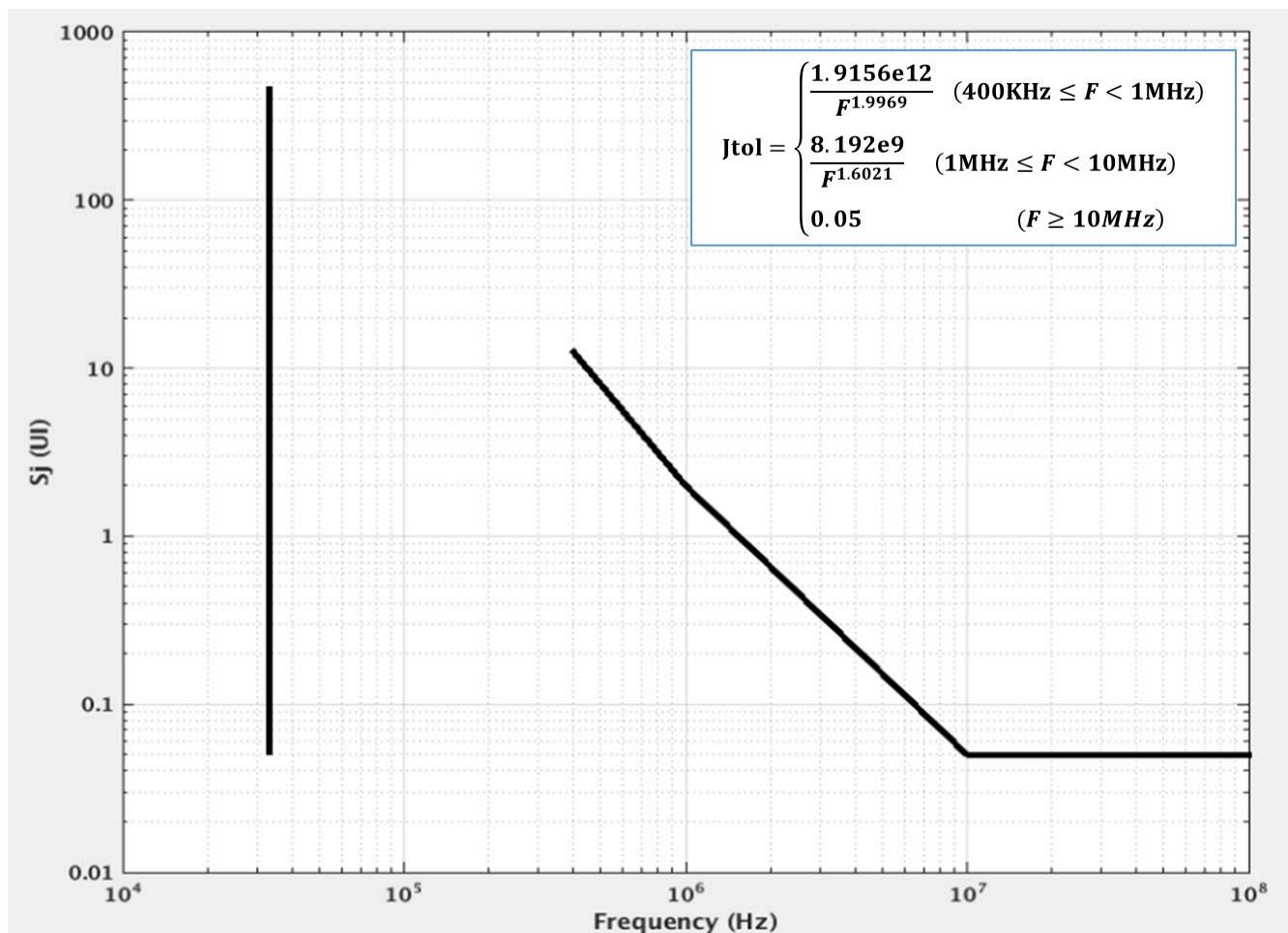


Figure 8-50 S_j Mask for Receivers Operating in SRIS mode at 64.0 GT/s §

Receivers operating at 64.0 GT/s in the CC Refclk mode use the S_j mask profile shown in § Figure 8-51 . The magnitude of the 33 kHz spur is 1 ns pp, or 32 UIpp. The equation of the swept S_j curve is shown on the § Figure 8-51 .

Base 6.4 vs Base 6.3

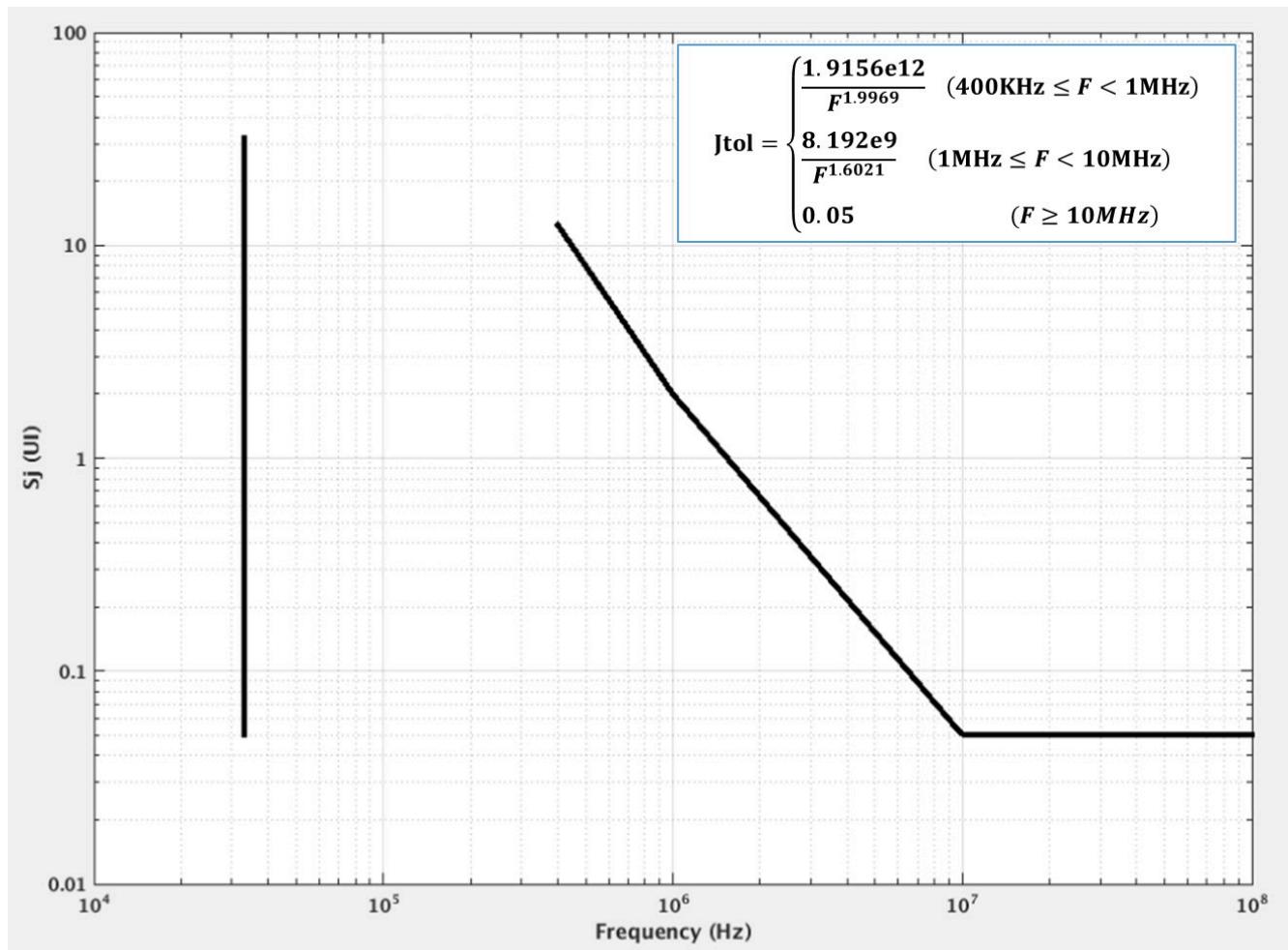


Figure 8-51 *S_j* Mask for Receivers Operating in CC mode at 64.0 GT/s §

Receivers operating in the CC Refclk mode at 8.0 GT/s shall utilize the *S_j* profile shown in § Figure 8-52 . The testing procedure is identical to that used for the IR mode, except that the clock topology differs. See § Section 8.4.2.3 for details. Receivers operating at 16.0 GT/s in the CC Refclk mode in devices that do not support 32.0 GT/s also have the option to use the *S_j* mask profile shown in § Figure 8-52 , with additional residual SSC applied per § Table 8-11 .

Base 6.4 vs Base 6.3

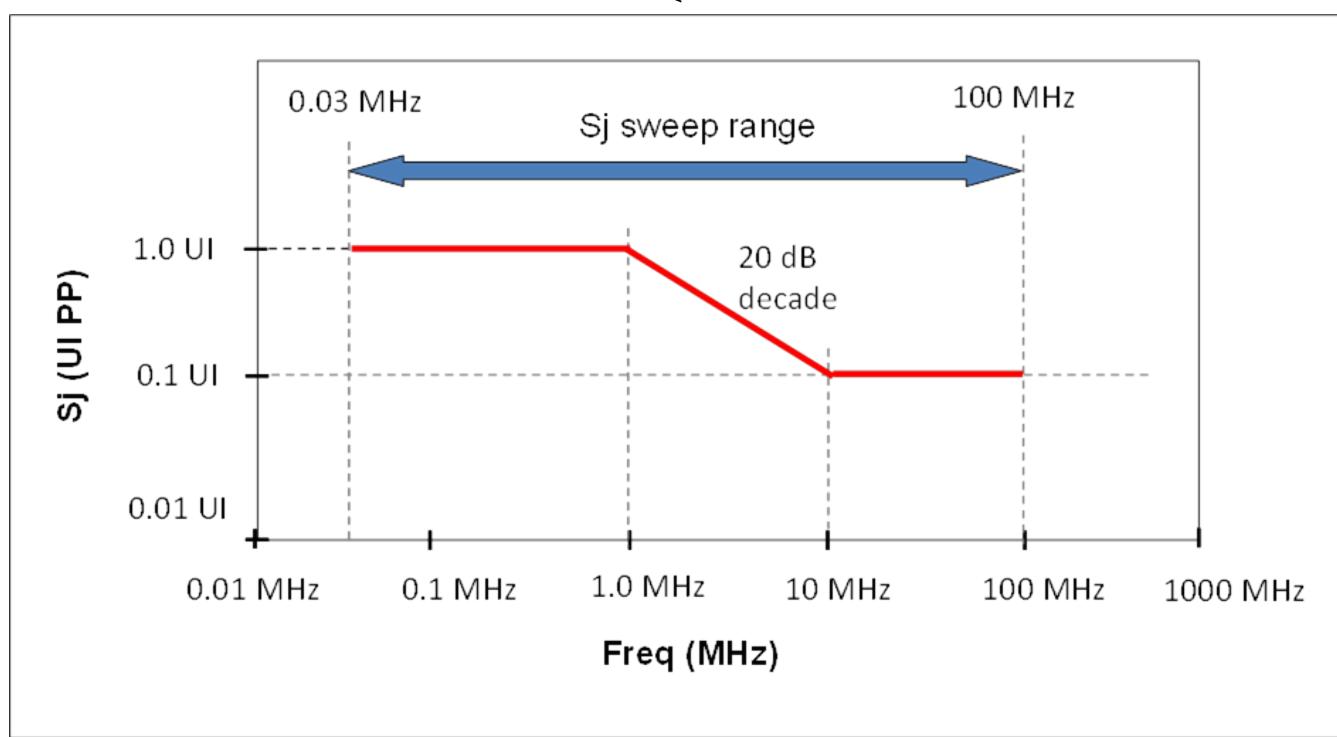


Figure 8-52 Sj Masks for Receivers Operating in CC Mode at 8.0 GT/s §

8.4.2.3 Receiver Refclk Modes §

A Rx is permitted to support one or both of two clock modes: CC and IR although only one clock mode may be operational at a given time. Receivers can support more than one Refclk mode by selecting a mode at power-up or by means of strapping pins, etc.

8.4.2.3.1 Common Refclk Mode §

§ Figure 8-53 shows the Refclk connection for a receiver in the Common Clock Refclk mode. A single Refclk source drives both the Generator and the DUT. This test utilizes the Sj mask specified in § Section 8.4.2.2.1 .

Base 6.4 vs Base 6.3

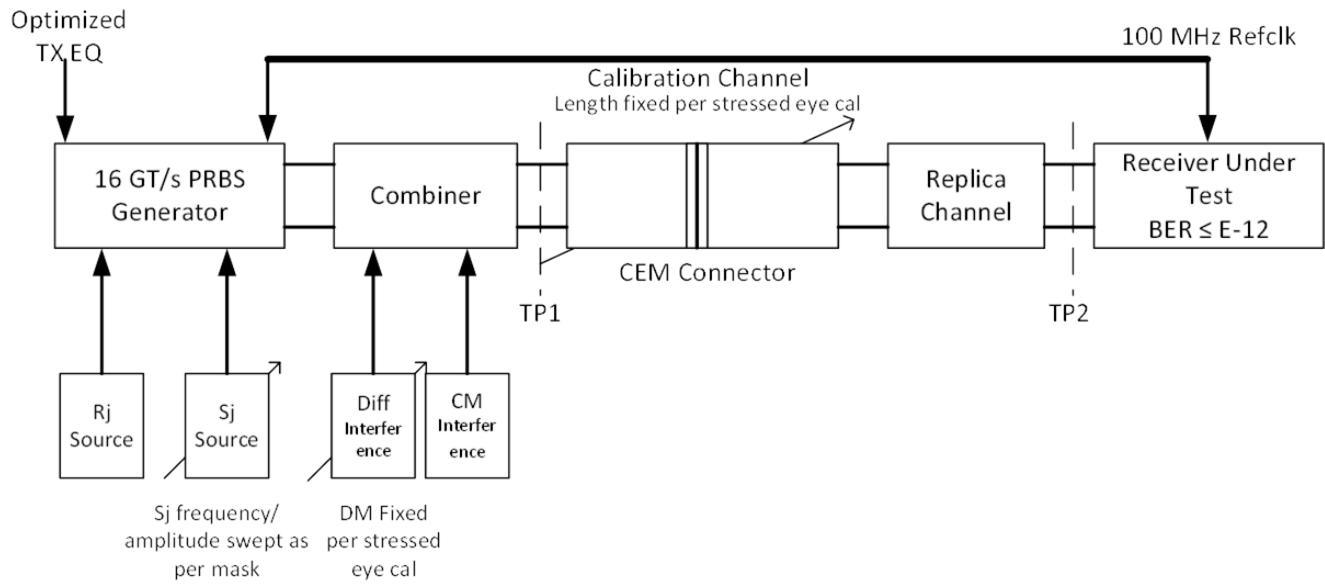


Figure 8-53 Layout for Jitter Testing Common Refclk Rx at 16.0 GT/s §

8.4.2.3.2 Independent Refclk Mode §

§ Figure 8-54 illustrates the configuration for testing a Receiver in the IR Refclk mode. A Refclk source with SSC is required for the DUT. The test utilizes the Sj mask specified in § Section 8.4.2.2.1 . The generator must be able to produce a large 33 KHz Sj tone while Sj is swept as shown in § Section 8.4.2.2.1 .

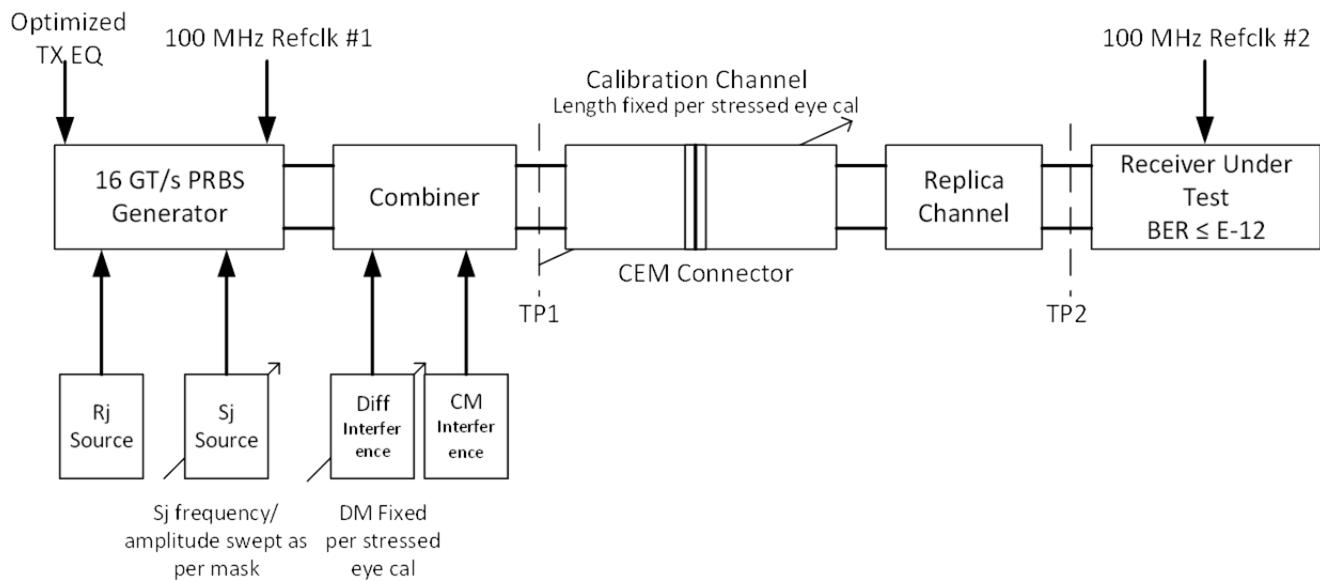


Figure 8-54 Layout for Jitter Testing for Independent Refclk Rx at 16.0 GT/s §

8.4.3 Common Receiver Parameters §

§ Table 8-12 lists the common Receiver parameters that are not directly associated with stressed eye tolerancing. Values are separately defined for the four data rates.

Table 8-12 Common Receiver Parameters §

| Symbol | Parameter | 2.5 GT/s value | 5.0 GT/s value | 8.0 GT/s value | 16.0 GT/s value | 32.0 GT/s value | 64.0 GT/s value | Units | Notes |
|--|---|---|---|---|---|---|---|-------|---|
| <i>UI (Rx)</i> | Unit Interval | (min) 399.88 (max) 400.12 (300 PPM) | (min) 199.94 (max) 200.06 (300 PPM) | (min) 124.9625 (max) 125.0375 (300 PPM) | (min) 62.48125 (max) 62.51875 (300 PPM) | (min) 31.246875 (max) 31.253125 (100 PPM) | (min) 31.246875 (max) 31.253125 (100 PPM) | ps | UI tolerance does not include SSC effects |
| <i>BW_{RX-PKG-PLL1}</i> | Rx PLL bandwidth corresponding to <i>PKG_{RX-PLL1}</i> | (max) 22 (min) 1.5 | (max) 16.0 (min) 8 | (max) 4.0 (min) 0.5 | (max) 4.0 (min) 0.5 | (max) 1.8 (min) 0.5 | (max) 1.0 (min) 0.5 | MHz | Second order PLL transfer bounding function. See Note 1. |
| <i>BW_{RX-PKG-PLL2}</i> | Rx PLL bandwidth corresponding to <i>PKG_{RX-PLL2}</i> | Not Specified | (max) 16.0 (min) 5.0 | (max) 5.0 (min) 0.5 | (max) 5.0 (min) 0.5 | N/A | N/A | MHz | Second order PLL transfer bounding function. See Note 1. |
| <i>PKG_{RX-PLL1}</i> | Maximum Rx PLL peaking corresponding to <i>BW_{RX-PKG-PLL1}</i> | (max) 3.0 | 3.0 | 2.0 | 2.0 | 2.0 | 2.0 | dB | Second order PLL transfer bounding function. See Note 1. |
| <i>PKG_{RX-PLL2}</i> | Maximum Rx PLL peaking corresponding to <i>BW_{RX-PKG-PLL2}</i> | Not specified | 1.0 | 1.0 | 1.0 | N/A | N/A | dB | Second order PLL transfer bounding function. See Note 1. |
| <i>RL_{RX-DIFF}</i> | Differential receiver return loss | See § Figure 8-24 | | | | | See § Figure 8-26 | dB | Note 2 |
| <i>RL_{RX-CM}</i> | Common mode receiver return loss | See § Figure 8-25 | | | | | See § Figure 8-27 | dB | Note 2 |
| <i>T_{RX-GND-FLOAT}</i> | Rx termination float time | | | (max) 500 | (max) 500 | (max) 500 | (max) 500 | μs | Note 5 |
| <i>V_{RX-CM-AC-P}</i> | Rx AC common Mode Voltage | (max) 150 | (max) 150 | (max) 75 | (max) 75 | (max) 75 | (max) 37.5 | mVP | Measured at Rx pins into a pair of 50 Ω terminations to ground |
| <i>Z_{RX-DC}</i> | Receiver DC single ended impedance | (min) 40 (max) 60 | (min) 40 (max) 60 | Not specified | Not specified | Not specified | Not specified | Ω | DC impedance limits are needed to guarantee Receiver detect. For 8.0, |

Base 6.4 vs Base 6.3

| Symbol | Parameter | 2.5 GT/s value | 5.0 GT/s value | 8.0 GT/s value | 16.0 GT/s value | 32.0 GT/s value | 64.0 GT/s value | Units | Notes |
|----------------------------------|---|--|--|--|--|--|--|----------|---|
| | | | | | | | | | 16.0, 32.0, and 64.0 GT/s it is bounded by $R_{L_{RX-CM}}$. See Note 3. |
| $Z_{RX-HIGH-IMP-DC-POS}$ | DC input CM input impedance for $V \geq 0$ during Reset or power-down | $\geq 10K$ (0-200 mV) $\geq 20K$ (> 200 mV) | Ω | Voltage measured wrt. ground. Parameters may not scale with process technology. See Note 4. |
| $Z_{RX-HIGH-IMP-DC-NEG}$ | DC input CM input impedance for $V < 0$ during Reset or power-down | 1.0K (min) | Ω | Voltage measured over the range of -150 mV to 0 mV wrt. ground. Parameters may not scale with process technology. See Note 4. |
| $V_{RX-IDLE-DET-DIFF-PP}$ | Electrical Idle Detect threshold | (min) 65 (max) 175 | (min) 65 (max) 175 | (min) 65 (max) 175 | (min) 65 (max) 175 | (min) 65 (max) 175 | (min) 65 (max) 175 | mV | $V_{RX-IDLE-DET-DIFFp-p} = 2 \times V_{RX-D+} - V_{RX-D-} $ |
| $T_{RX-IDLE-DET-DIFF-ENTERTIME}$ | Unexpected Electrical Idle Enter Detect Threshold Integration Time | (max) 10 | (max) 10 | (max) 10 | (max) 10 | (max) 10 | (max) 10 | ms | An unexpected Electrical Idle ($V_{RX-DIFF-PP} < V_{RX-IDLE-DET-DIFFp-p}$) must be recognized no longer than $T_{RX-IDLE-DET-DIFF-ENTERTIME}$ to signal an unexpected idle condition. |
| $L_{RX-SKEW}$ | Lane to Lane skew | (max) 20 | (max) 8 | (max) 6 | (max) 5 | (max) 5 | (max) 5 | ns | Across all Lanes on a Port. $L_{RX-SKEW}$ comprehends Lane-Lane variations due to channel and repeater delay differences. |

Notes:

- Two combinations of PLL BW and peaking are specified at 5.0 GT/s to permit designers to make tradeoffs between the two parameters. If the PLL's min BW is ≥ 8 MHz, then up to 3.0 dB of peaking is permitted. If the PLL's min BW is relaxed to ≥ 5.0 MHz, then a tighter peaking value of 1.0 dB must be met. Note: a PLL BW extends from zero up to the value(s) defined as the min or max in the above table. For 2.5 GT/s a single PLL bandwidth and peaking value of 1.5-22 MHz and 3.0 dB are defined.
- Measurements must be made for both common mode and differential return loss. In both cases the DUT must be powered up and DC isolated, and its D+/D- inputs must be in the low-Z state.

| Symbol | Parameter | 2.5 GT/s value | 5.0 GT/s value | 8.0 GT/s value | 16.0 GT/s value | 32.0 GT/s value | 64.0 GT/s value | Units | Notes |
|--------|--|----------------|----------------|----------------|-----------------|-----------------|-----------------|-------|-------|
| 3. | The Rx DC single ended impedance must be present when the Receiver terminations are first enabled to ensure that the Receiver Detect occurs properly. Compensation of this impedance is permitted to start immediately and the Rx Common Mode Impedance (constrained by RL_{RX-CM} to $50 \Omega \pm 20\%$) must be within the specified range by the time Detect is entered. | | | | | | | | |

4. $Z_{RX-HIGH-IMP-DC-NEG}$ and $Z_{RX-HIGH-IMP-DC-POS}$ are defined respectively for negative and positive voltages at the input of the Receiver. Transmitter designers need to comprehend the large difference between >0 and <0 Rx impedances when designing Receiver detect circuits.

5. Defines the time for the Receiver's input pads to settle to new common-mode on 2.5/5.0 GT/s transition to 8.0, 16.0, 32.0, and 64.0 GT/s.

8.4.3.1 5.0 GT/s Exit From Idle Detect (EFI) §

It is difficult to scale the capabilities of the EFI detect circuits with data rate, and for this reason the 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s specification defines different data patterns in the FTS and the TS1 and TS2 Ordered Sets than are defined for 2.5 GT/s operation. In particular, repeated K28.7 patterns are defined to guarantee sufficient voltage and time requirements, as illustrated in the figure below. Concatenated EIE Symbols yield alternating one/zero run lengths of five UI each.

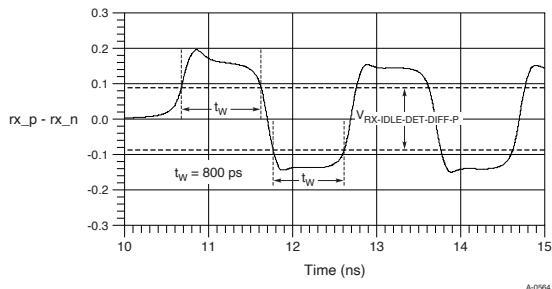


Figure 8-55 Exit from Idle Voltage and Time Margins §

8.4.3.2 Receiver Return Loss §

The measurement methodology and frequency binning for differential and common mode Rx RL is identical to that for the Tx. For details refer to § Figure 8-24 , § Figure 8-26 , § Figure 8-25 , and § Figure 8-27 .

8.4.4 Lane Margining at the Receiver - Electrical Requirements §

PCI Express components including Retimers that support the 16.0 GT/s or higher rate are required to support Lane margining at the Receiver when operating at 16.0 GT/s or higher. Lane Margining enables system software to get the margin information of a given Lane while the Link is in L0 state. For NRZ signaling, the margin information includes both voltage and time, in either direction from the current Receiver position. For PAM4 signaling, the margin information includes both voltage and time for all the three eyes. The margin feature is not permitted to require any additional external hardware to function. Support of Lane margining for voltage is optional at 16.0 GT/s and required at 32.0 GT/s or higher. Support of independent timing margin to the left or to the right is optional for all data rates. Support of independent voltage margin for up or for down is optional for all data rates that support voltage margining. For simplicity, the margin commands and requirements described in § Section 4.2.18 are described in terms of moving the

data sample location - but the actual margining method is implementation specific. For example - the timing margin could be implemented on the actual data sampler or an independent error sampler. Further the timing margin can be achieved by injecting an appropriate amount of stress/jitter to the data sample location, or by moving the data/error sample location. The parameters in § Table 8-13 are reported for 16.0, 32.0, or 64.0 GT/s and all are allowed to be different for each of those data rates, though some are required to be different for NRZ vs. PAM4 data rates.

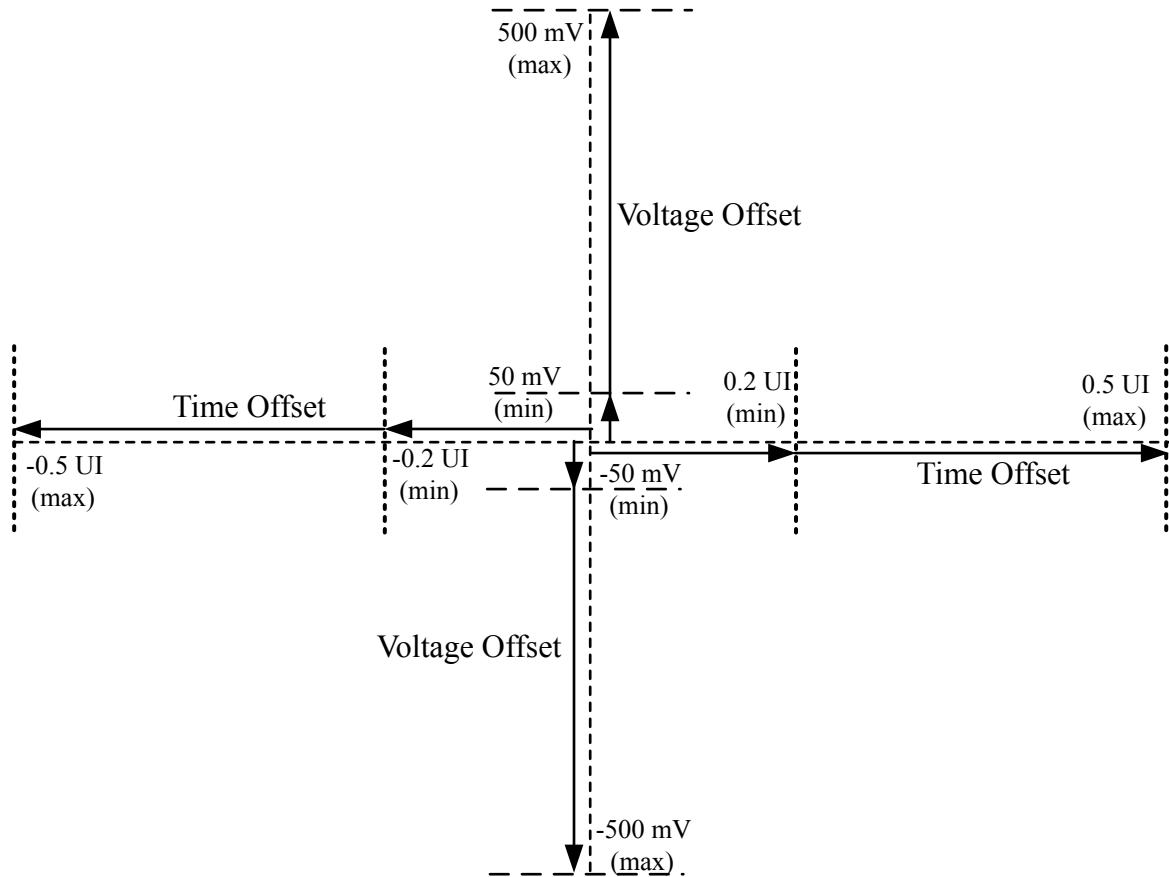


Figure 8-56 Allowed Ranges for Maximum NRZ Timing and Voltage Margin §

Base 6.4 vs Base 6.3

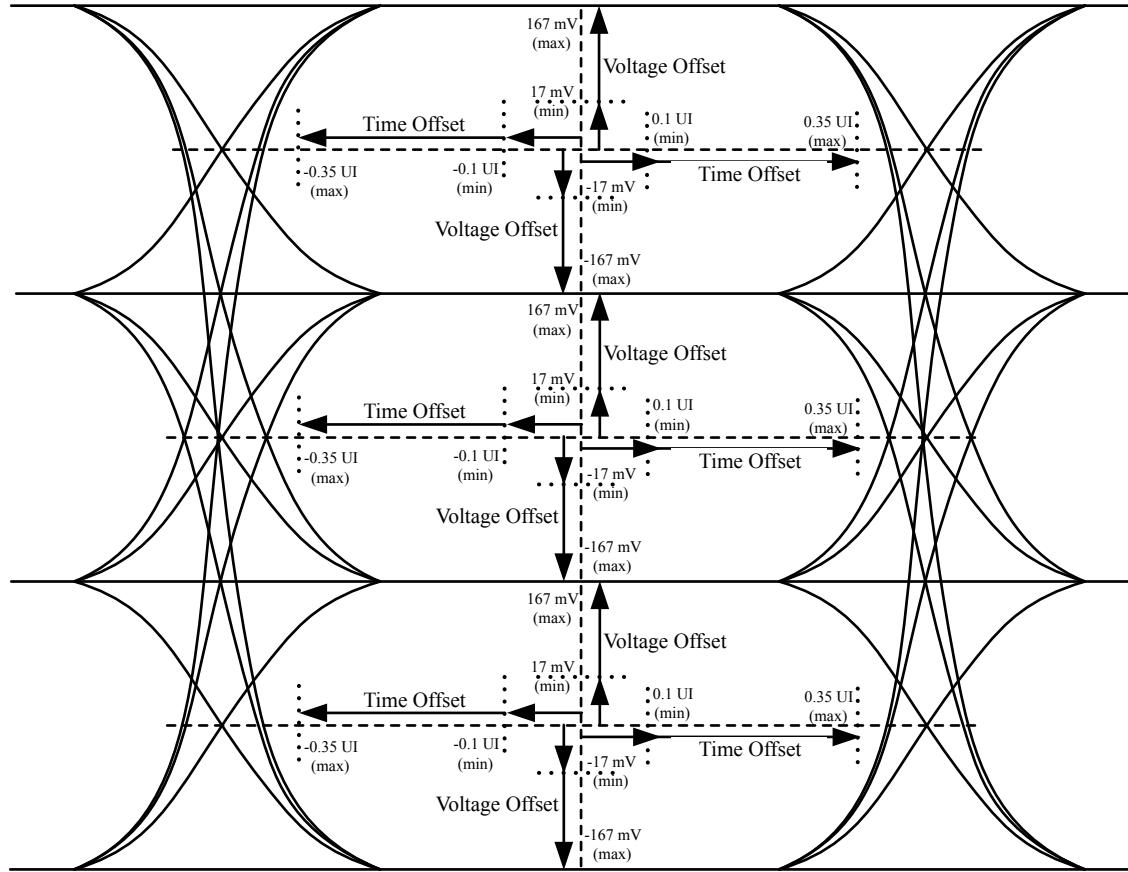


Figure 8-57 Allowed Ranges for Maximum PAM4 Timing and Voltage Margins [§](#)

Table 8-13 Lane Margining [§](#)

| Parameter Name | Min | Max | Description |
|-----------------------|---------------------------|---------------------------|--|
| $M_{NumTimingSteps}$ | 6 (NRZ) 10 (PAM4) | 63 | <p>Number of time steps from default (to either left or right), range must be at least $+/-0.2$ UI if operating in NRZ, or at least $+/-0.1$ UI if operating in PAM4</p> <p>Timing offset must increase monotonically</p> <p>The number of steps in both positive (toward the end of the unit interval) and negative (toward the beginning of the unit interval) must be identical</p> |
| $M_{MaxTimingOffset}$ | 20 % (NRZ) 10 % (PAM4) | 50 % (NRZ) 35 % (PAM4) | <p>Offset from default at maximum step value as percentage of a nominal UI</p> <p>A 0 value may be reported if the vendor chooses not to report the offset</p> |
| $M_{NumVoltageSteps}$ | 32 (NRZ) 64 (PAM4) | 127 | <p>Number of voltage steps from default (either up or down), minimum range $+/-50$ mV if operating in NRZ, or $+/-17$ mV if operating in PAM4, either as measured by the reference equalizer</p> <p>Voltage offset must increase monotonically</p> |

| Parameter Name | Min | Max | Description |
|--------------------------|-----|------|--|
| | | | <p>The number of steps in both positive and negative direction from the default sample location must be identical</p> <p>This value is undefined if <u>MVoltageSupported</u> is 0b</p> |
| MMaxVoltageOffset | 5 % | 50 % | <p>Offset from default at maximum step value as percentage of one volt if operating in NRZ, or as a percentage of one third vold if operating in PAM4</p> <p>A 0 value may be reported if the vendor chooses not to report the offset when <u>MVoltageSupported</u> is 1b</p> <p>This value is undefined if <u>MVoltageSupported</u> is 0b</p> |

Placeholder table row so Errata B677 is properly attributed in changebar Base 6.3 documents. Remove this table row for subsequent revisions.

Errata B677 combined:

- $M_{NumTimingStepsPAM4}$ into MNumTimingSteps
- $M_{MaxTimingOffsetPAM4}$ into MMaxTimingOffset
- $M_{NumVoltageStepsPAM4}$ into MNumVoltageSteps
- $M_{MaxVoltageOffsetPAM4}$ into MMaxVoltageOffset

| | | | |
|-----------------------------|---|----|--|
| MSamplingRateVoltage | 0 | 63 | The ratio of bits tested to bits received during voltage margining. A value of 0 is a ratio of 1:64 (1 bit of every 64 bits received), and a value of 63 is a ratio of 64:64 (all bits received). |
| MSamplingRateTiming | 0 | 63 | The ratio of bits tested to bits received during timing margining. A value of 0 is a ratio of 1:64 (1 bit of every 64 bits received), and a value of 63 is a ratio of 64:64 (all bits received). |
| MVoltageSupported | 0 | 1 | 1b indicates that voltage margining is supported |
| MIndLeftRightTiming | 0 | 1 | 1b indicates independent left/right timing margin supported |
| MIndUpDownVoltage | 0 | 1 | 1b independent up and down voltage margining supported |
| MIndErrorSampler | 0 | 1 | <p>1b Margining will not produce errors (change in the error rate) in data stream (i.e., error sampler is independent)</p> <p>0b Margining may produce errors in the data stream</p> |
| MMaxLanes | 0 | 31 | <p>Maximum number of Lanes minus 1 that can be margined at the same time. It is recommended that this value be greater than or equal to the number of Lanes in the Link minus 1. Encoding Behavior is undefined if software attempts to margin more than <u>MMaxLanes</u> +1 at the same time.</p> <p>Note: This value is permitted to exceed the number of Lanes in the Link minus 1.</p> |

| Parameter Name | Min | Max | Description |
|--------------------------------|-----|-----|---|
| <i>M_SampleReportingMethod</i> | 0 | 1 | Indicates whether sampling rates (<i>M_SamplingRateVoltage</i> and <i>M_SamplingRateTiming</i>) are supported (1) or a sample count is supported (0). One of the two methods is supported by each device. |
| <i>M_ErrorCount</i> | 0 | 63 | If <i>M_IndErrorSampler</i> is 1b this is a count of the actual bit errors since margining started. If <i>M_IndErrorSampler</i> is 0b this is the actual count of the logical errors since margining started. See the Physical Layer Logical Block chapter for the definition of what errors are counted. The count saturates at 63. |
| <i>M_SampleCount</i> | 0 | 127 | Value = $3 \cdot \log_2$ (number of bits margined). Where number of bits margined is a count of the actual number of bits tested during margining. The count stops when margining stops. The count saturates at 127 (after approximately 5.54×10^{12} bits). The count resets to zero when a new margin command is received. |

8.4.5 Low Frequency and Miscellaneous Signaling Requirements §

8.4.5.1 ESD Standards §

All PCI Express signal and power supply pins must be tested for ESD protection levels to the Human Body Model (HBM) and the Charged Device Model (CDM) standards in accordance with [ESDA-JEDEC-JS-001-2010] (for HBM) and in accordance with [JEDEC-JESD22-C101] (for CDM). Pins must meet or exceed the minimum levels recommended in [JEDEC-JEP155-JEP157] (HBM/CDM) or JEDEC approved superseding documents.

8.4.5.2 Channel AC Coupling Capacitors §

Each Lane of a PCI Express Link must be AC coupled. The minimum and maximum values for the capacitance are given in § Table 8-7 . Capacitors must be placed on the Transmitter side of an interface that permits adapters to be plugged and unplugged. In a topology where everything is located on a single substrate, the capacitors may be located anywhere along the channel. External capacitors are assumed because the values required are too large to feasibly construct on-chip.

8.4.5.3 Short Circuit Requirements §

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

8.4.5.4 Transmitter and Receiver Termination §

- The Transmitter is required to meet $RL_{TX-DIFF}$ and RL_{TX-CM} (see § Figure 8-24 , § Figure 8-26 , § Figure 8-25 and § Figure 8-27) any time functional differential signals are being transmitted.
- The Transmitter is required only to meet $I_{TX-SHORT}$ (see § Table 8-7 any time functional differential signals are not being transmitted.
- Note: The differential impedance during this same time is not defined.
- The Receiver is required to meet $RL_{RX-DIFF}$ and RL_{RX-CM} (see § Table 8-12) during all LTSSM states excluding only times during when the device is powered down, Fundamental Reset is asserted, or when explicitly specified.
- The Receiver is required to meet $Z_{RX-HIGH-IMP-DC-NEG}$ and $Z_{RX-HIGH-IMP-DC-POS}$ (see § Table 8-12) any time adequate power is not provided to the Receiver, Fundamental Reset is asserted, or when explicitly specified.

8.4.5.5 Electrical Idle §

Electrical Idle is a steady state condition where the Transmitter D+ and D- voltages are held constant at the same value. Electrical Idle is primarily used in power saving and inactive states (e.g., Disabled).

Before a Transmitter enters Electrical Idle, it must always send the required number of EIOSs except for the LTSSM substates explicitly exempted from this requirement. After sending the last Symbol of the last of the required number of EIOSs, the Transmitter must be in a valid Electrical Idle state within the time as specified by $T_{TX-IDLE-SET-TO-IDLE}$ in § Table 8-7 .

The successful reception of an EIOS occurs based on the rules defined in the Physical Layer Logical Block chapter. It should be noted that in substates (e.g., Loopback.Active for a Loopback Follower) where multiple consecutive EIOSs are expected, the Receiver must receive the appropriate number of EIOS sequences comprising of COM, IDL, IDL, IDL.

The low impedance common mode and differential Receiver termination values (see § Table 8-7 and § Table 8-12) must be met in Electrical Idle. The Transmitter can be in either a low or high impedance mode during Electrical Idle.

Any time a Transmitter enters Electrical Idle it must remain in Electrical Idle for a minimum of $T_{TX-IDLE-MIN}$. The Receiver should expect the last EIOS followed by a minimum amount of time in Electrical Idle ($T_{TX-IDLE-MIN}$) to arm its Electrical Idle Exit detector.

When the Transmitter transitions from Electrical Idle to a valid differential signal level it must meet the output return loss specifications described in § Figure 8-24 , § Figure 8-26 , § Figure 8-25 , and § Figure 8-27 .

Electrical Idle Exit shall not occur if a signal smaller than $V_{RX-IDLE-DET-DIFFp-p}$ minimum is detected at a Receiver. Electrical Idle Exit shall occur if a signal larger than $V_{RX-IDLE-DET-DIFFp-p}$ maximum is detected at a Receiver. Electrical Idle may be detected on the received signal regardless of its frequency components, or it may be detected only when the received signal is switching at a frequency of 125 MHz or higher.

8.4.5.6 DC Common Mode Voltage §

The Receiver DC common mode voltage is nominally 0 V when operating at 2.5 GT/s or 5.0 GT/s.

Transmitter DC common mode voltage is held at the same value during all states unless otherwise specified. The range of allowed Transmitter DC common mode values is specified in § Table 8-7 ($V_{TX-DC-CM}$).

8.4.5.7 Receiver Detection §

The Receiver Detection circuit is implemented as part of a Transmitter and must correctly detect whether a load impedance equivalent to a DC impedance implied by the Z_{RX-DC} parameter ($40\ \Omega$ - $60\ \Omega$) is present. Note: Support for Rx detect, which only occurs at 2.5 GT/s, is the reason why 2.5 GT/s Receivers impedance at DC is specified.

The recommended behavior of the Receiver Detection sequence is described below:

- Step 1. A Transmitter must start at a stable voltage prior to the detect common mode shift.
- Step 2. A Transmitter changes the common mode voltage on D+ and D- consistent with meeting the $V_{TX-RCV-DETECT}$ parameter and consistent with detection of Receiver high impedance which is bounded by parameters $Z_{RX-HIGH-IMP-DC-POS}$, $Z_{RX-HIGH-IMP-DC-NEG}$ in § Table 8-12. Receiver is detected based on the rate that the lines change to the new voltage.
- The Receiver is not present if the voltage at the Transmitter charges at a rate dictated only by the Transmitter impedance and the capacitance of the interconnect and series capacitor.
 - 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.

If the Receiver Detection circuit performs the detect sequence on each conductor of the differential pair (both D+ and D-) and detects a load impedance greater than Z_{RX-DC} on either conductor, the Receiver Detection circuit shall interpret this as no termination load present and respond as if neither load were present.

It is required that the detect sequence be performed on both conductors of a differential pair.

8.5 Channel Tolerancing §

8.5.1 Channel Compliance Testing §

This section of the specification is relevant only for those cases where a platform design comprehends the relevant channel between Transmitter device pins and Receiver device pins. These types of platform designs are called “captive channels”. Designs that are not captive channels shall refer to the appropriate form factor (CEM is one example) specification, since in this case the form factor specification takes precedence over this specification and splits the channel between two different types of components.

The key components and processes of channel tolerancing are illustrated in § Figure 8-58 and § Figure 8-59. The major difference lies in the complexity of the Behavioral Tx and Rx equalization, which depends on the data rate. 2.5 and 5.0 GT/s utilize fixed Tx presets and assume no Rx equalization, whereas 8.0, 16.0, 32.0, and 64.0 GT/s assume multi-valued, adjustable Tx presets and a combination of Rx DFE and CTLE.

Base 6.4 vs Base 6.3

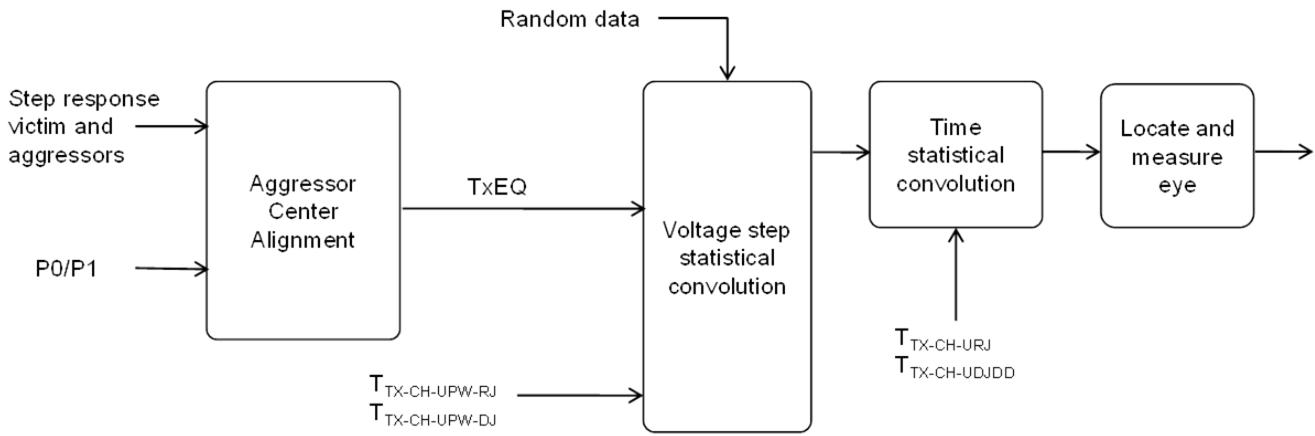


Figure 8-58 Flow Diagram for Channel Tolerancing at 2.5 and 5.0 GT/s §

The basic channel compliance approach is to first acquire the channel's characteristics, usually by means of s-parameters or equivalent model. Behavioral Tx and Rx package models are then appended to the channel model to yield a die pad to die pad topology. The model shall include both victim path and a sufficient number of aggressor paths to accurately capture channel crosstalk effects. Using the Tx voltage and jitter limits defined in the Transmitter specification section it is possible to transform these parameters to what would appear at the die pad of a Tx.

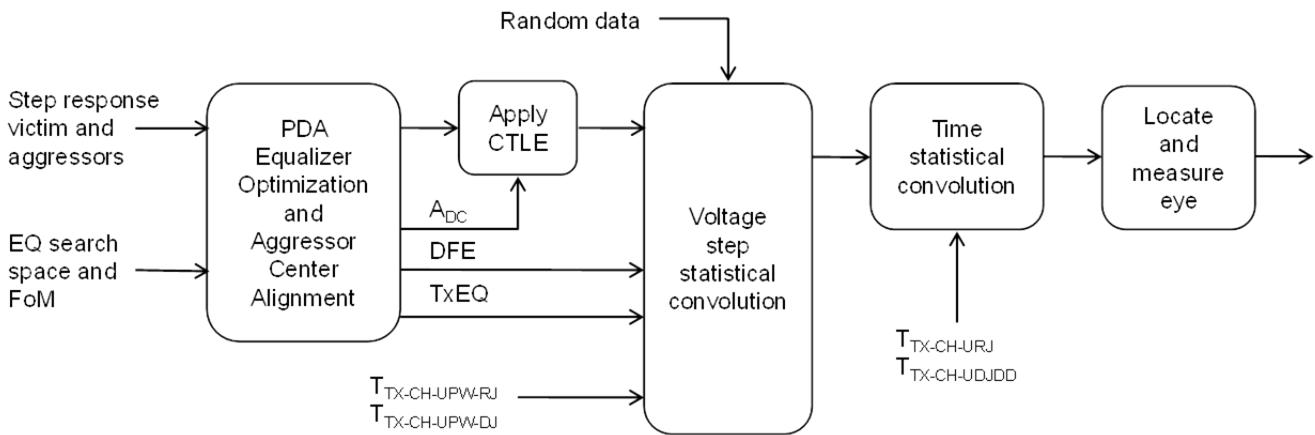


Figure 8-59 Flow Diagram for Channel Tolerancing at 8.0 and 16.0 GT/s §

The resulting model is analyzed via simulation, yielding voltage and jitter at a point equivalent to the input latch of the Receiver. The signal observed at the Receiver's latch is referenced to a recovered data clock from which an eye diagram may be constructed.

For 8.0, 16.0, 32.0, and 64.0 GT/s testing the simulation process must properly account for Tx and Rx equalization optimization as must be supported by a minimum capability Tx/Rx pair. This means the simulation process must be able to select the optimum values for the Tx presets or coefficients and Rx equalization settings based upon:

- a 1st order CTLE and a 1-tap DFE for 8.0 GT/s,
- a 1st order CTLE and a 2-tap DFE for 16.0 GT/s,

- a 2nd order CTLE and a 3-tap DFE for 32.0 GT/s, and
- a CTLE transfer function with six poles and three zeros and a 16-tap DFE for 64.0 GT/s.

8.5.1.1 Behavioral Transmitter and Receiver Package Models §

Behavioral package models are defined in this specification to represent the combined die and package loss that is expected to interoperate with the targeted range of channels. Note that at 16.0 GT/s, the behavioral packages represent a high, but not worst-case, loss for many devices. (see § Section 8.3.3.11 and § Section 8.4.1.5)

A separate pair of package models are defined for 8.0, 16.0, 32.0, and 64.0 GT/s. At 8.0 GT/s, separate package models are defined for TX and RX ports to reflect the smaller CPAD capacitance typical in most Receiver implementations. At 16.0, 32.0, and 64.0 GT/s, separate package models are defined for devices containing Root Ports and all other devices to reflect the large and socketed nature of most devices containing Root Complexes. Channel testing for all three data rates typically requires testing in both directions.

The package models are included with the specification as design collateral. Each model for 8.0 and 16.0 GT/s comprehends C_{PIN} and C_{PAD} parasitic capacitances plus a differential t-line element as illustrated in § Figure 8-60.

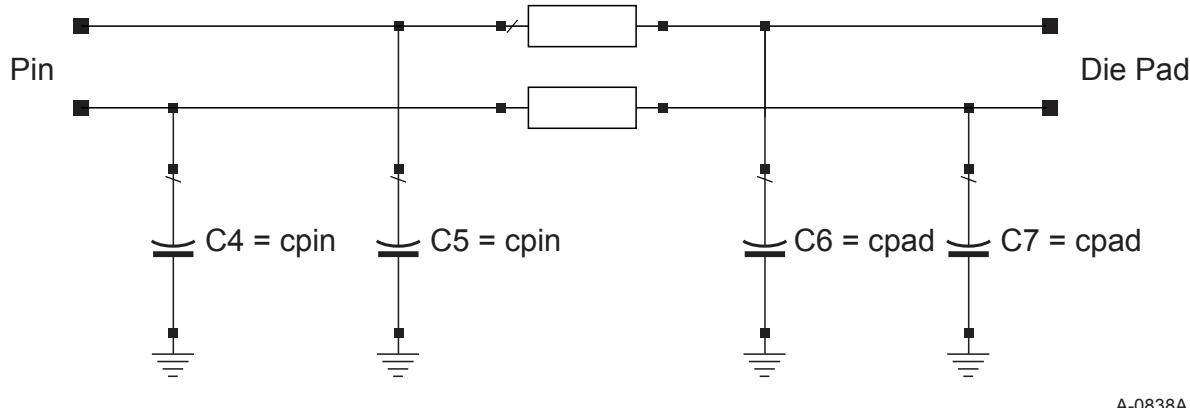


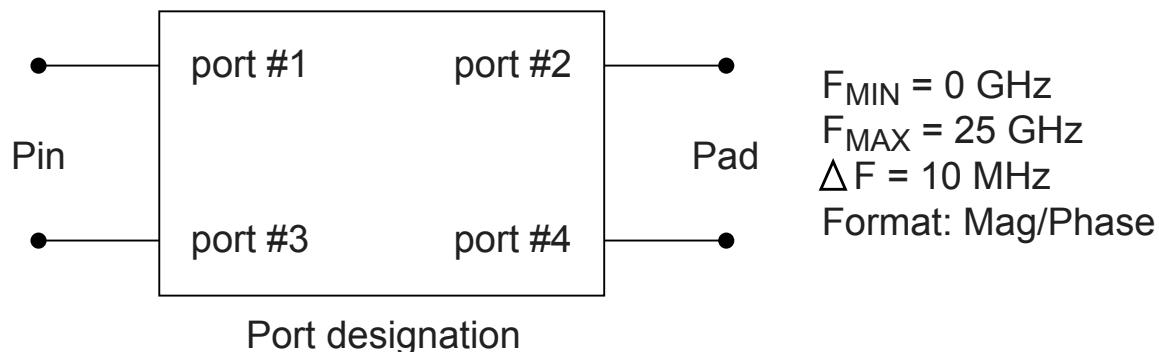
Figure 8-60 Tx/Rx Behavioral Package Models §

The C_{PIN} and C_{PAD} values used in the package model generation are provided for informative purposes only.

Table 8-14 Package Model Capacitance Values §

| | 8.0 GT/s Tx | 8.0 GT/s Rx | 16.0 GT/s |
|-----------|-------------|-------------|-----------|
| C_{PIN} | 0.25 pF | 0.25 pF | 0.25 pF |
| C_{PAD} | 1.0 pF | 0.8 pF | 0.5 pF |

For ease of incorporation into the post processing flow the 8.0 and 16.0 behavioral package models are specified as 4-port s-parameter files. The files are specified with port designations, frequency range and granularity as listed below. The reference impedance for the s-parameters is 50 Ω. File format is Touchstone.



A-0839A

Figure 8-61 Behavioral Tx and Rx S-Port Designation for 8.0 and 16.0 GT/s Packages §

Base 6.4 vs Base 6.3

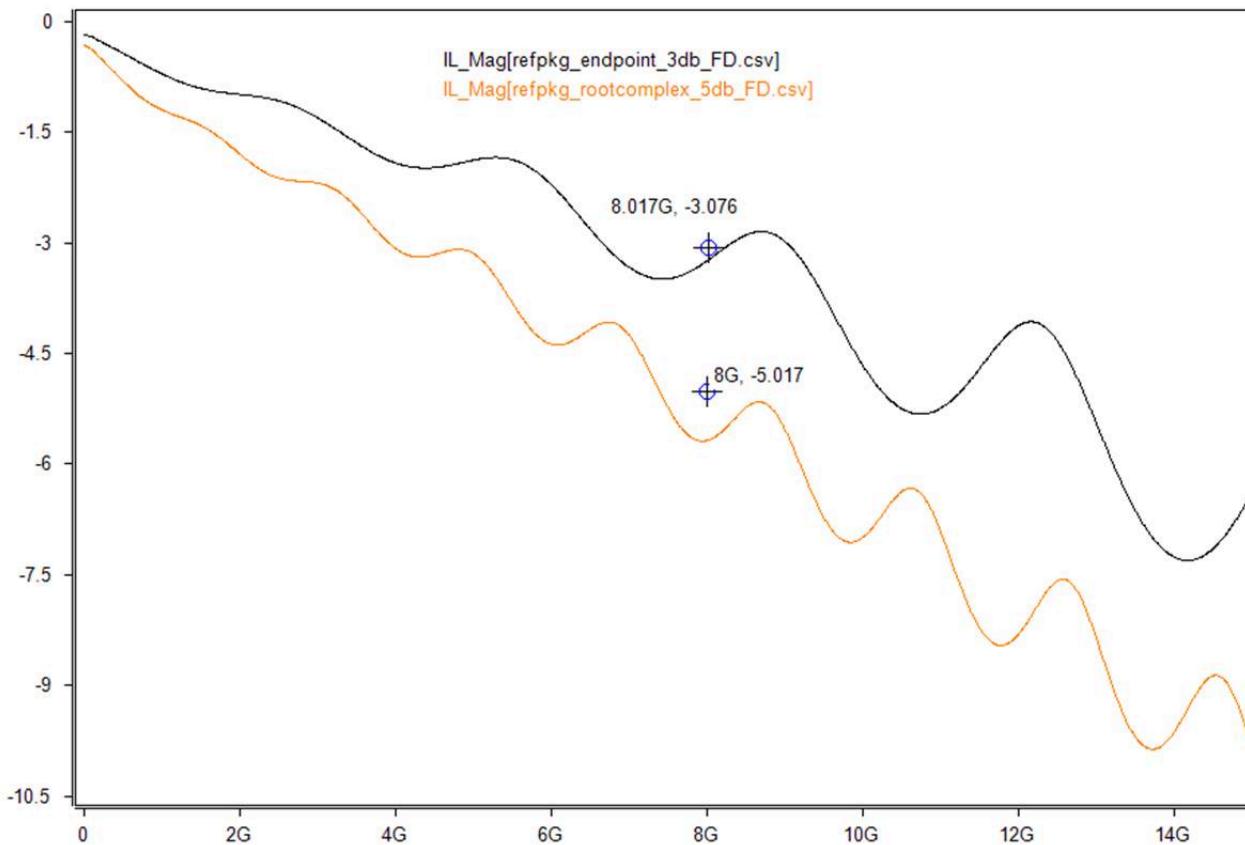


Figure 8-62 SDD21 Plots for Root and Non-Root Packages for 16.0 GT/s §

For 32.0 and 64.0 GT/s the package models are based on real package and socket models for the root package and package and BGA models for the non-root package.

For 32.0 GT/s, reference package models the die capacitive loads are included in the models. For 64.0 GT/s, the effective die cap has been replaced by the S-parameter representation of an on-die network that represents the insertion and

return loss characteristics of on-die pad for a typical design. The reference impedance for the 32.0 and 64.0 GT/s reference package model s-parameters is $50\ \Omega$. Figures below show 32.0 and 64.0 GT/s reference package insertion loss, return loss, FEXT, and NEXT S-parameter plots based on $50\ \Omega$ impedance. To account for the on-die inductive coil DC loss of about $3\ \Omega$, the Tx and Rx DC termination should be set to $47\ \Omega$ in 64.0 GT/s channel compliance simulations.

Base 6.4 vs Base 6.3

Insertion loss

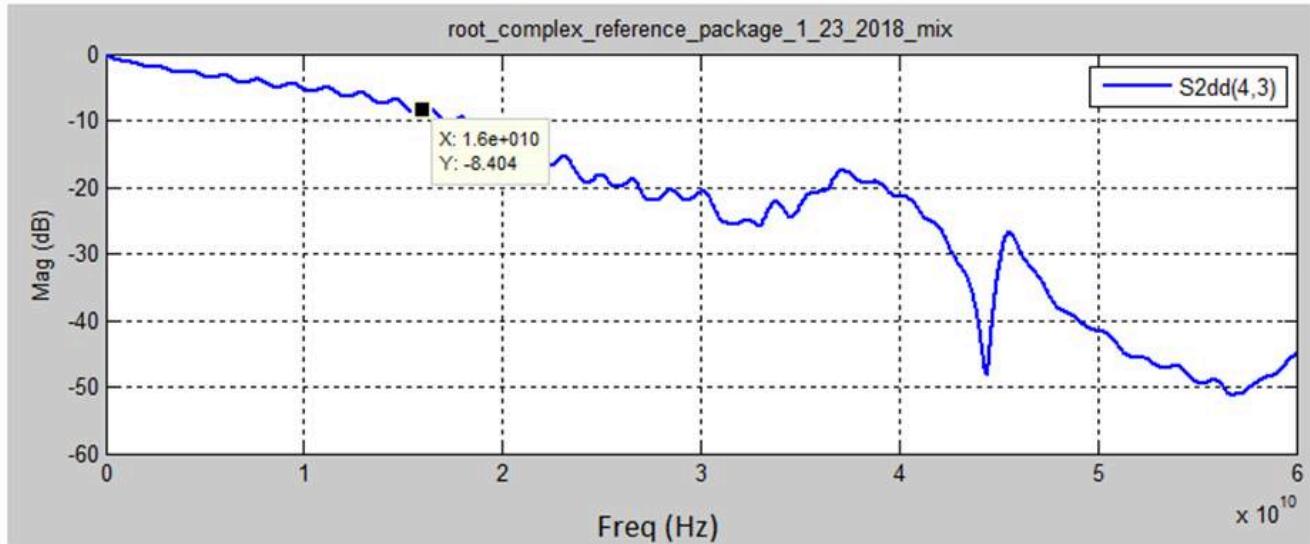


Figure 8-63 Insertion Loss for Root Reference Package for 32.0 GT/s [§](#)

Return Loss (Board-side)

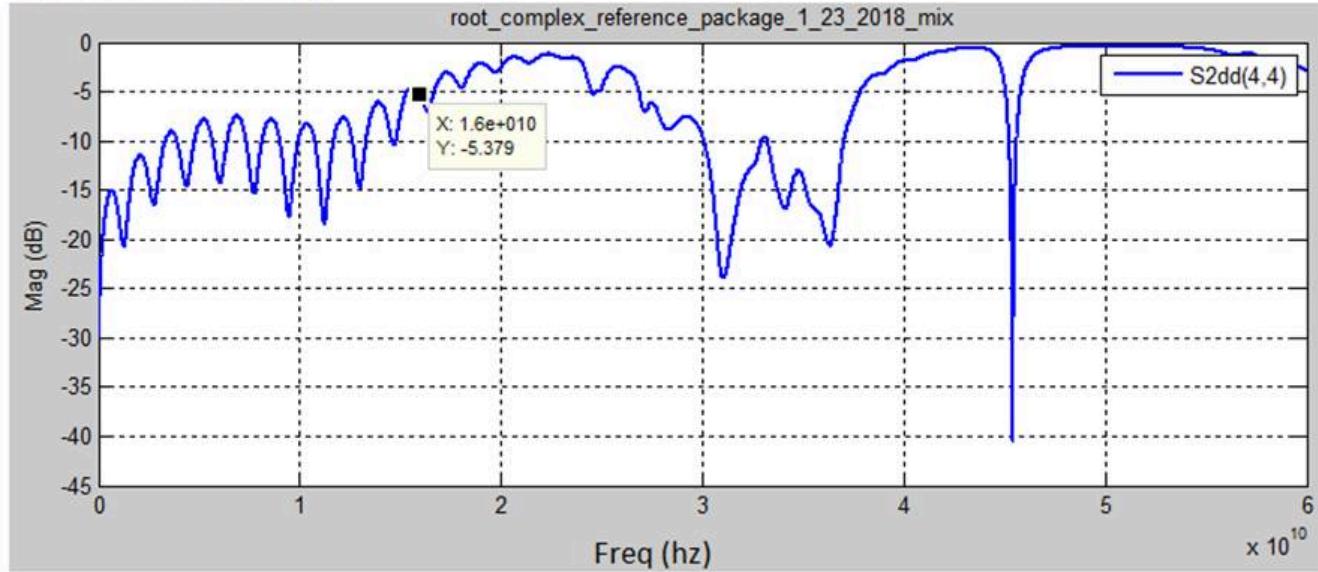


Figure 8-64 Return Loss for Root Reference Package for 32.0 GT/s [§](#)

Base 6.4 vs Base 6.3

NEXT 2

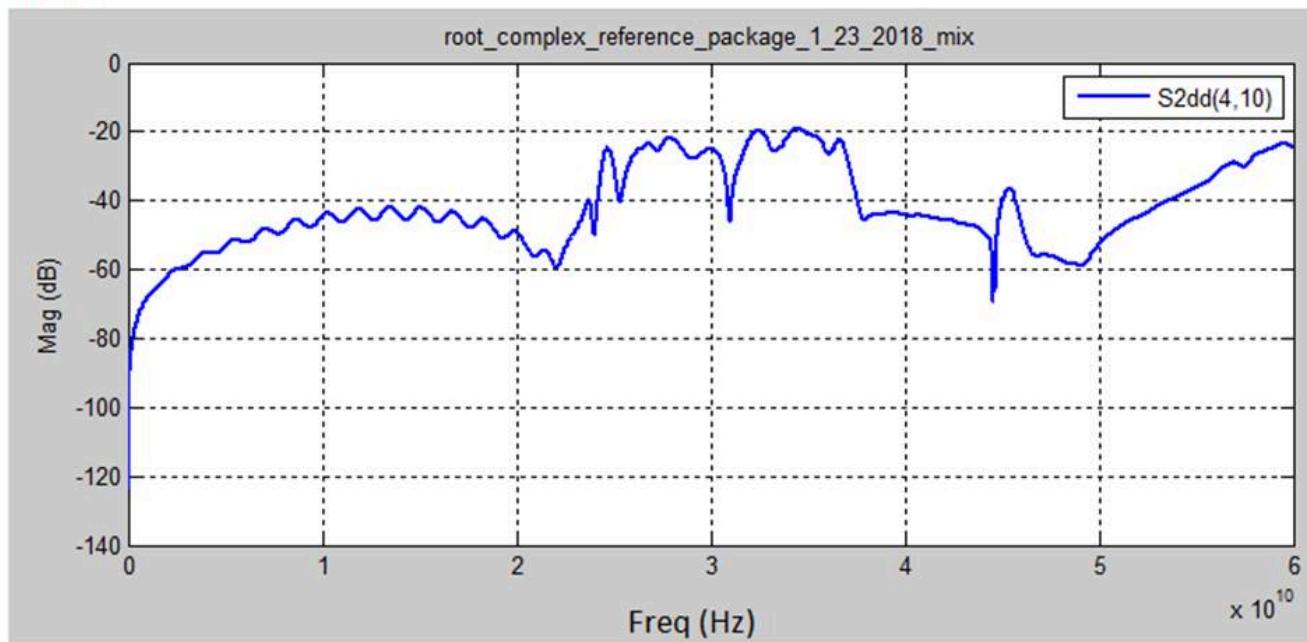


Figure 8-65 NEXT for Root Reference Package (Worst-Case) for 32.0 GT/s §

FEXT 1

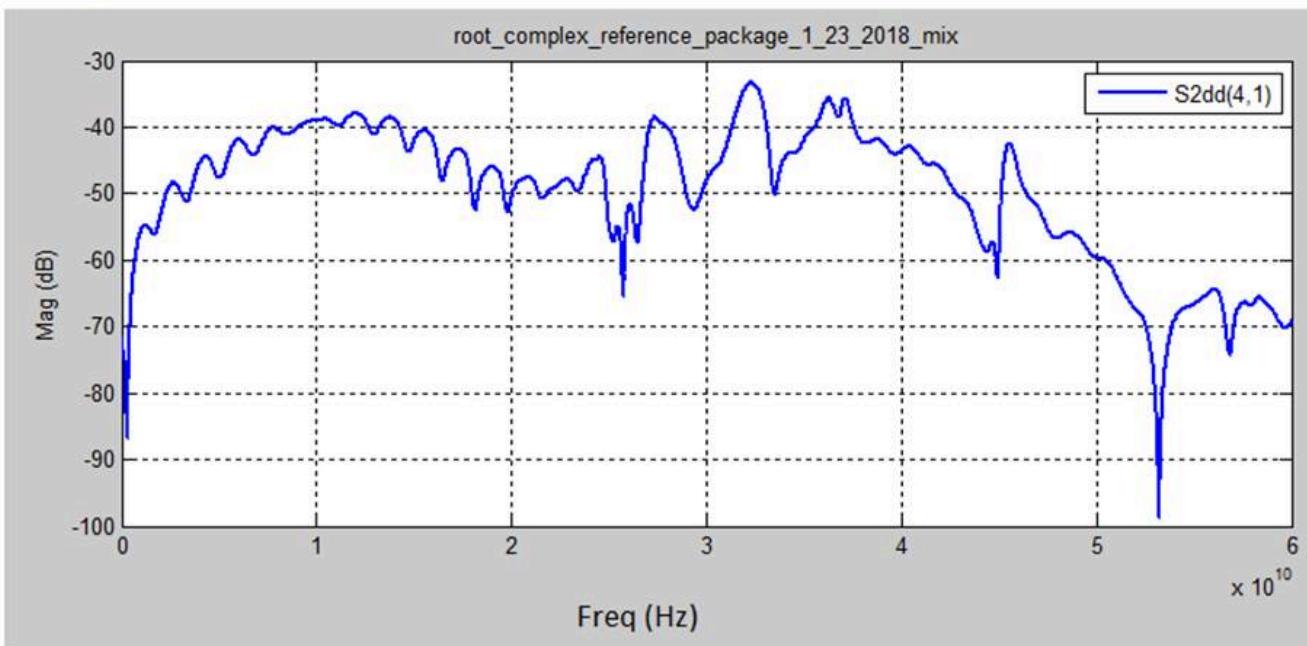


Figure 8-66 FEXT for Root Reference Package (Worst-Case) for 32.0 GT/s §

Base 6.4 vs Base 6.3

Insertion loss

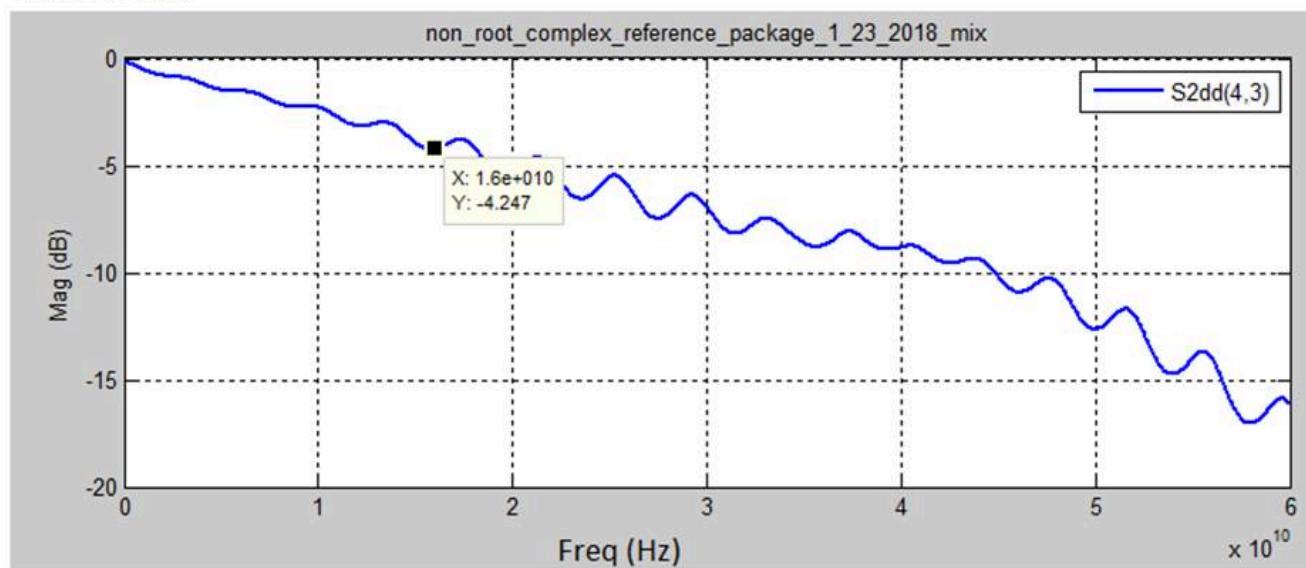


Figure 8-67 Insertion Loss for Non-Root Reference Package for 32.0 GT/s §

Return Loss (Board-side)

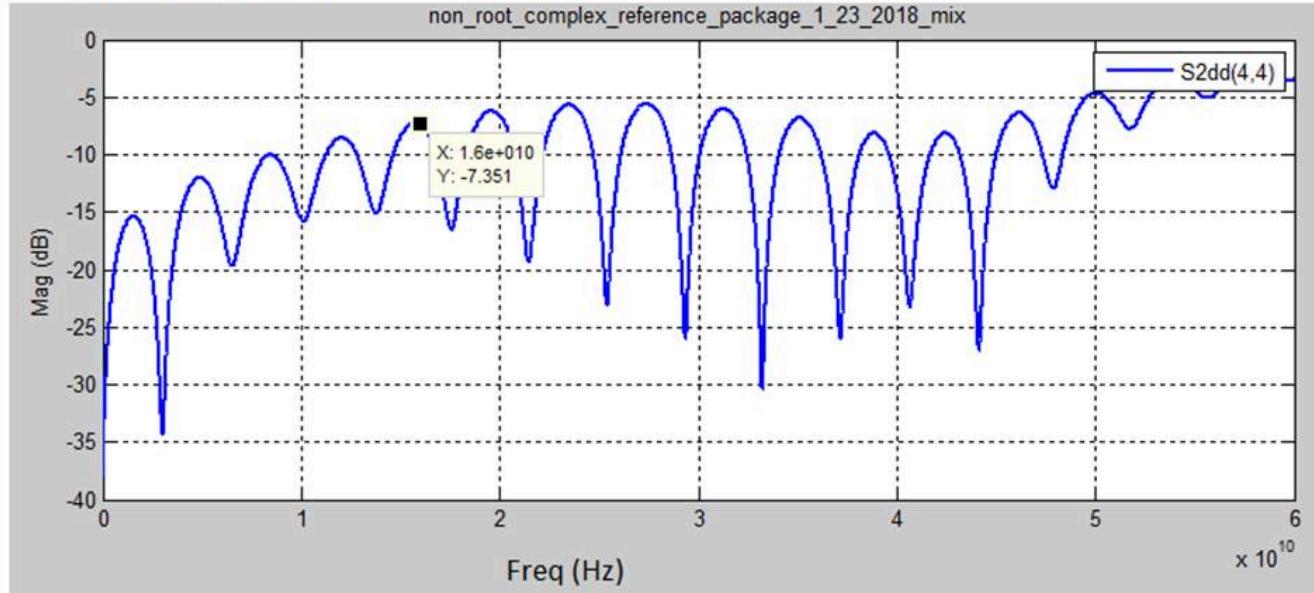


Figure 8-68 Return Loss for Non-Root Reference Package for 32.0 GT/s §

Base 6.4 vs Base 6.3

NEXT 1

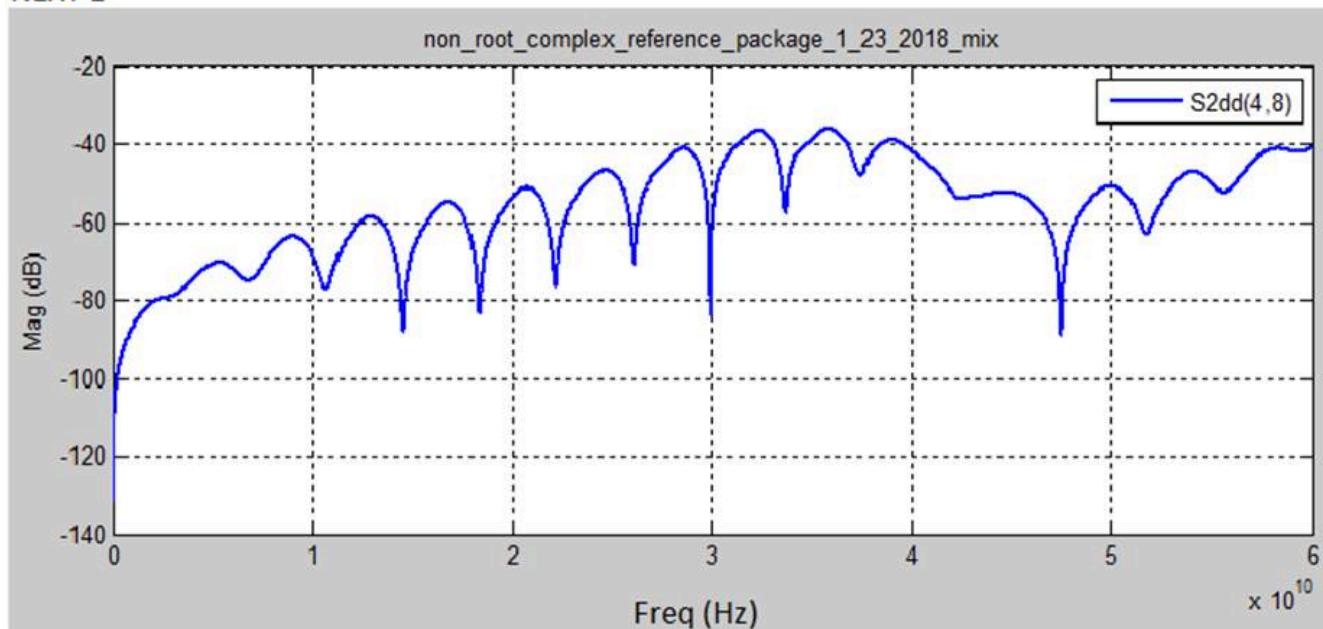


Figure 8-69 NEXT for Non-Root Reference Package (Worst-Case) for 32.0 GT/s §

FEXT 2 (Die-side)

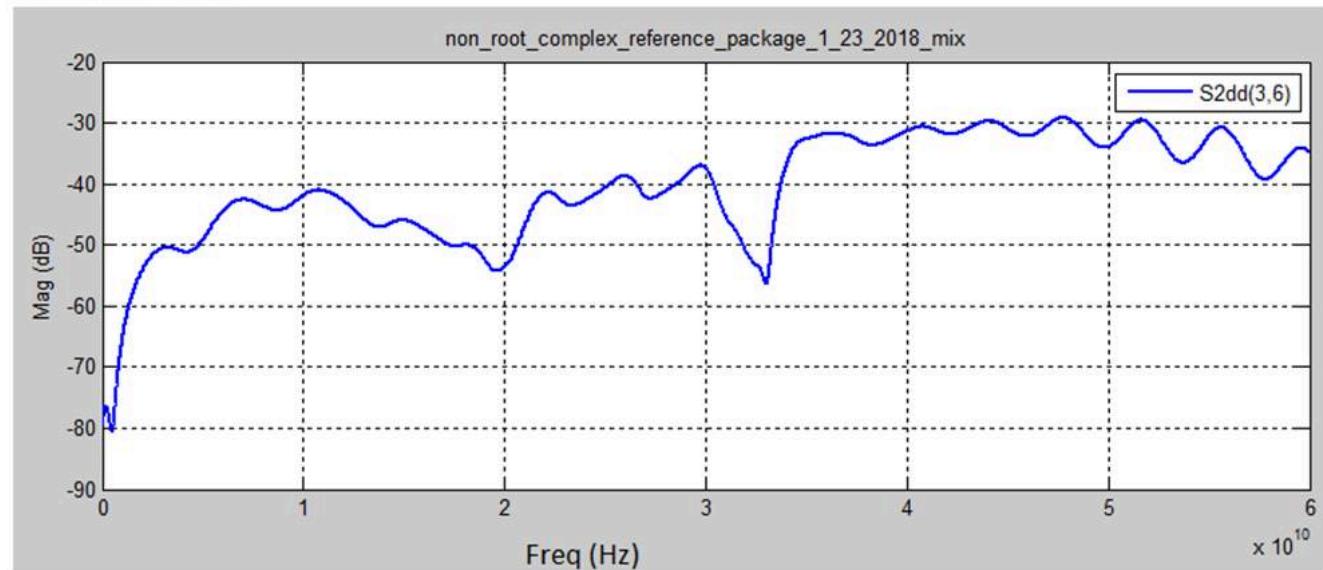


Figure 8-70 FEXT for Non-Root Reference Package (Worst-Case) for 32.0 GT/s §

Base 6.4 vs Base 6.3

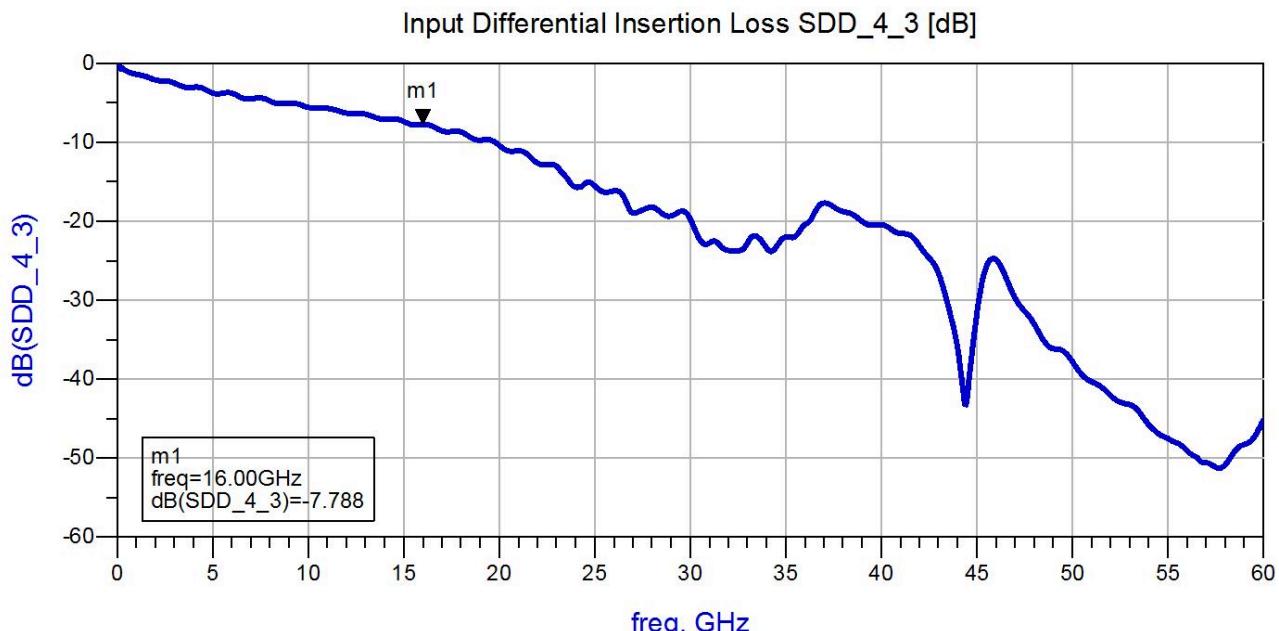


Figure 8-71 Insertion Loss for Root Reference Package for 64.0 GT/s §

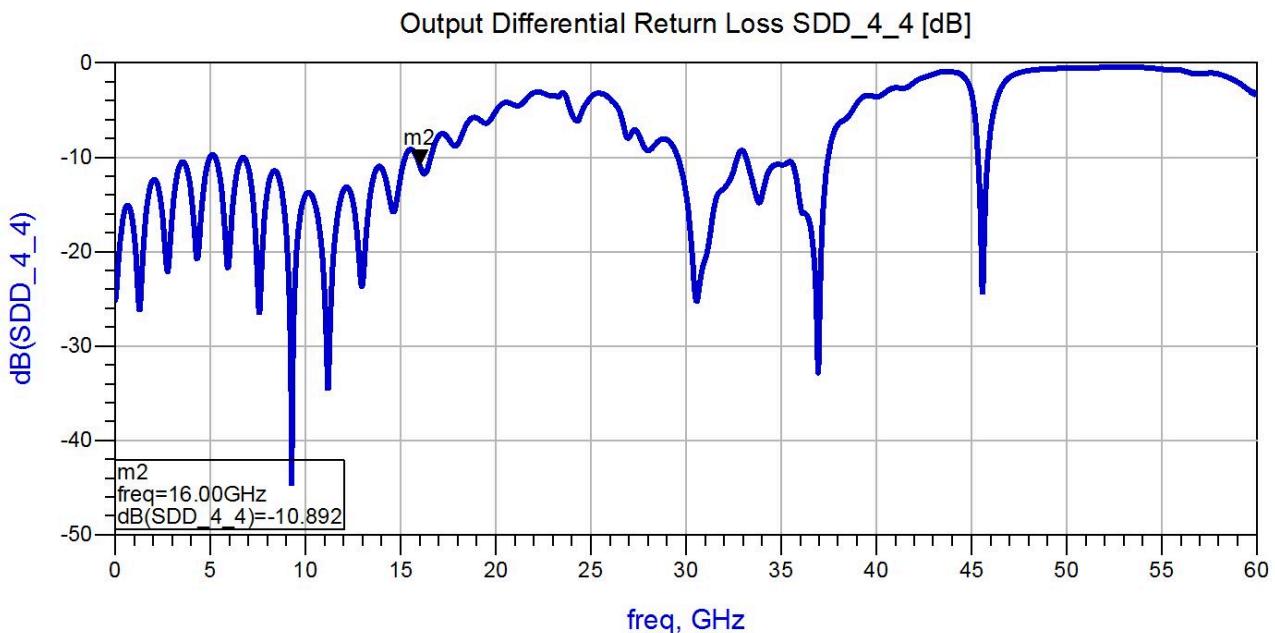


Figure 8-72 Return Loss for Root Reference Package for 64.0 GT/s §

Base 6.4 vs Base 6.3

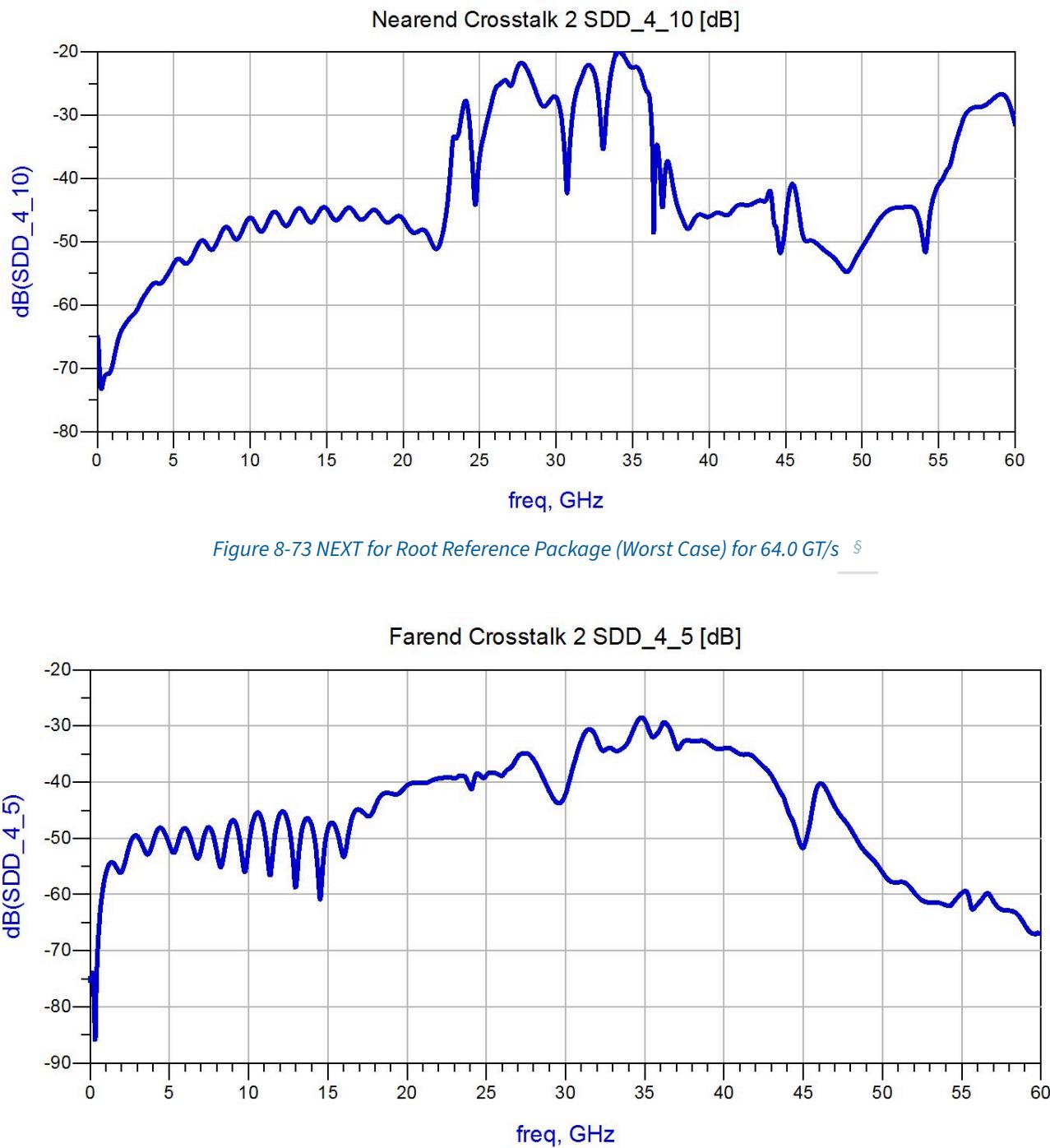


Figure 8-73 NEXT for Root Reference Package (Worst Case) for 64.0 GT/s §

Base 6.4 vs Base 6.3

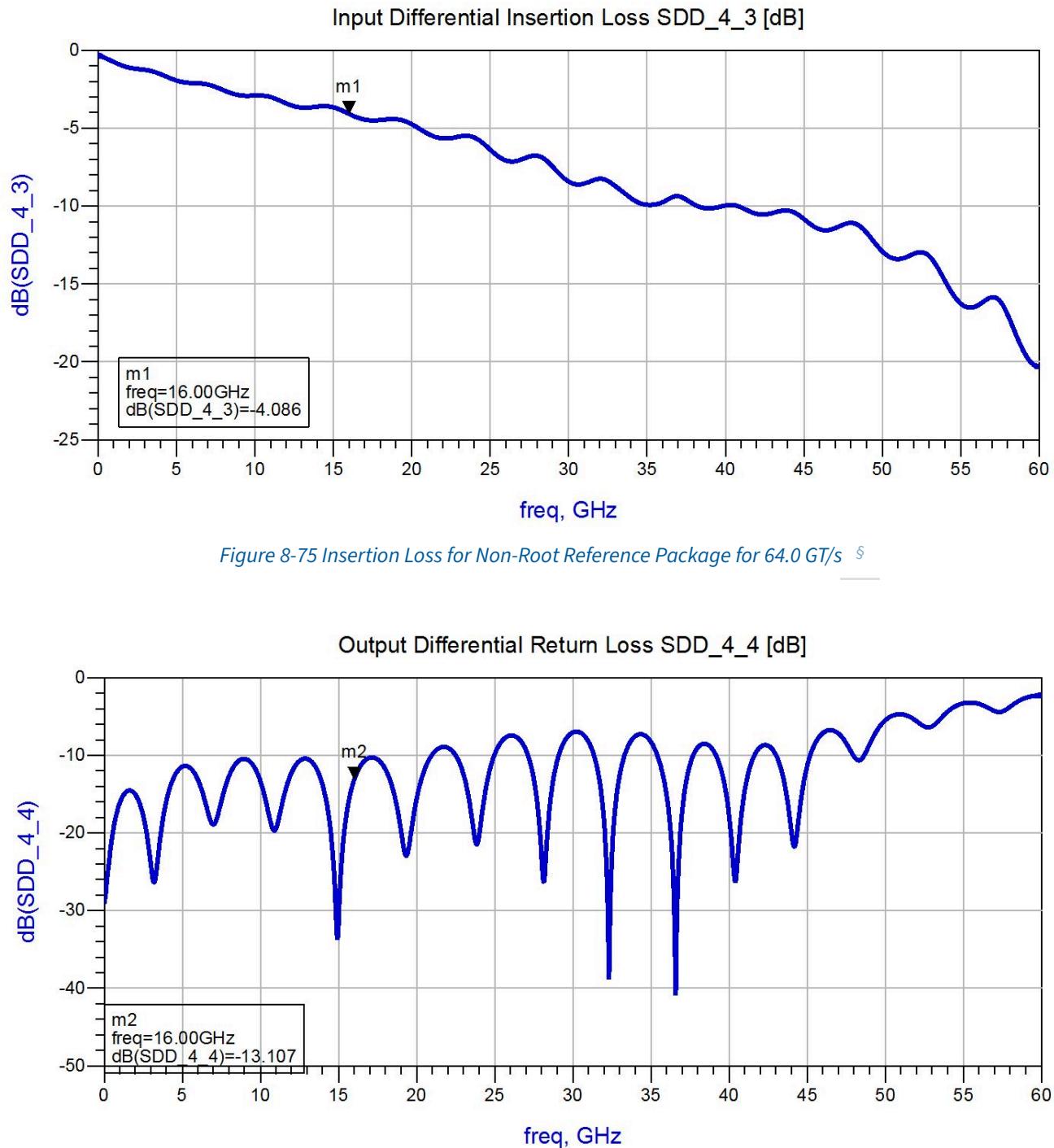


Figure 8-75 Insertion Loss for Non-Root Reference Package for 64.0 GT/s §

Base 6.4 vs Base 6.3

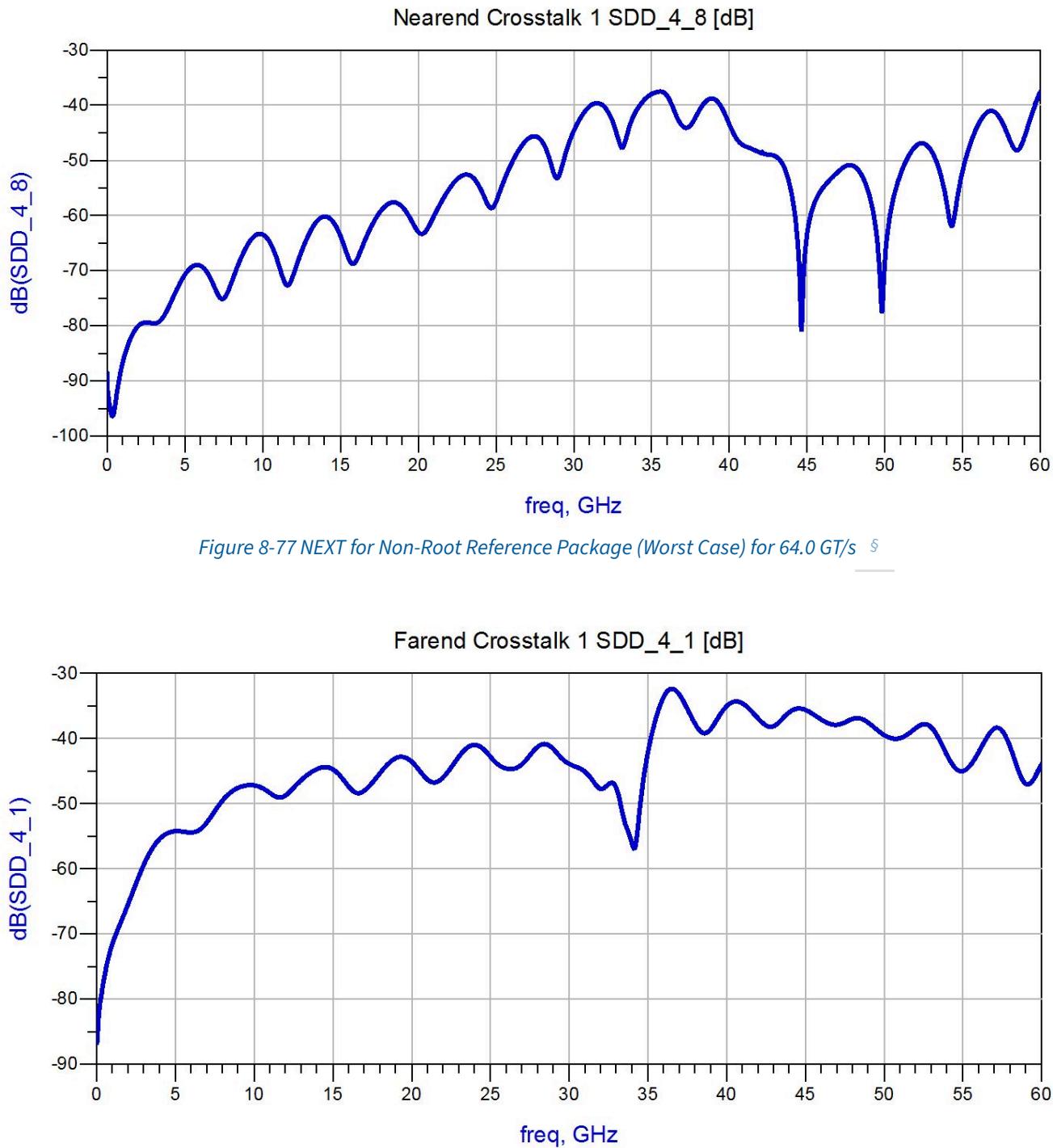


Figure 8-77 NEXT for Non-Root Reference Package (Worst Case) for 64.0 GT/s §

Figure 8-78 FEXT for Non-Root Reference Package (Worst Case) for 64.0 GT/s §

Base 6.4 vs Base 6.3

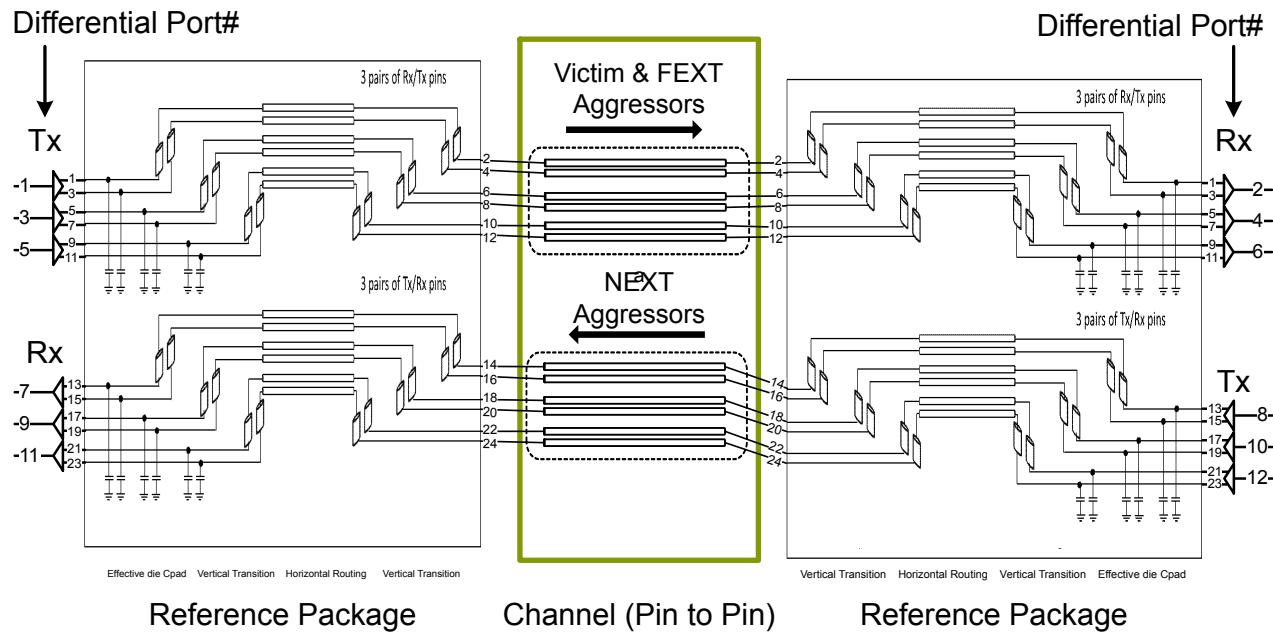


Figure 8-79 32.0 and 64.0 GT/s Reference Package Port Connections for Pin to Pin Channel Evaluation §

§ Figure 8-79 shows the port connections for using the 32.0 and 64.0 GT/s reference packages to evaluate a pin to pin channel using the channel compliance methodology. Both directions (root transmitting and non-root transmitting) must be evaluated. The lane labeled channel 3 to channel 4 is intended as the worst-case victim channel and must be evaluated in both directions. Other channels in the reference package model are intended only for use as crosstalk aggressors (not victim channels).

8.5.1.2 Measuring Package Performance (16.0 GT/s only) §

Package insertion loss at 16.0 GT/s is an informative spec parameter. Some implementations at 16.0 GT/s (see § Section 8.3.3.11) are allowed to have packages that exceed reference packages in insertion loss and/or crosstalk. Actual package performance must be assessed by performing channel compliance with reference channels provided with the specification on the PCI-SIG website. A set of channel compliance simulations are run on the reference channels with one of the reference packages being replaced by the package that is being evaluated. If the eye height or eye width is smaller for any of the channels with the package that is being evaluated, then the package is considered to have worse performance than the reference package. An implementation with a package that has worse performance than the reference package must use the implementation package model in the channel compliance methodology and may optionally use the implementation package in the Receiver stressed eye calibration. Note that form factor and channel compliance (for a captive channel) overall requirements still need to be met regardless of package characteristics.

8.5.1.3 Simulation Tool Requirements §

Channel tolerancing is implemented by means of simulation, where the pass/fail criteria are defined in terms of a time domain eye diagram. The simulation tool must accept a prescribed set of inputs, including the channel under

consideration and then simulate based upon a set of post processing requirements. This specification does not stipulate the use of any specific tool for simulating channels. However, any simulation tool must meet the following requirements.

8.5.1.3.1 Simulation Tool Chain Inputs §

- Channel characteristics defined as S-parameters or equivalent model. The model must include the victim differential Lane plus as many aggressors as required to accurately capture crosstalk. In most cases this will be between 2 and 4 additional differential Lanes. Note that 32.0 and 64.0 GT/s are most likely to require additional aggressors to accurately capture worst-case crosstalk and most likely to require several NEXT aggressors to capture crosstalk accurately.
- Behavioral Root and Non-Root package models. The models will be included as part of the specification in the form of s-parameter files (see § Section 8.5.1.1).
- Transmitter Jitter and voltage: The voltage and jitter parameters input to the simulator may be directly obtained from a combination of the Transmitter and Refclk jitter. Since these parameters are fixed the simulation tool may choose to hard code their values.
- Transmitter and Receiver Termination Impedance: The simulator shall use a $2 \times 50 \Omega$ termination for both the Transmitter and Receiver. This value matches the assumptions which are implicit in generating and measuring the stressed eye for Rx tolerancing.

8.5.1.3.2 Processing Steps §

- Time domain representation of the end-to-end connectivity: Included are the behavioral Tx and Rx packages and the channel under test.
- Tx voltage and jitter: Voltage and jitter parameters defined for the Transmitter but have been recalculated to properly comprehend high and low frequency jitter components, and also include Refclk jitter contributions.
- Behavioral Transmitter Equalization: The simulator shall replicate the Transmitter equalization capabilities defined in the Transmitter section.
- Behavioral Rx CTLE: The simulation tool shall implement a behavioral CTLE that replicates the CTLE function employed for Rx tolerancing.
- Behavioral DFE: The simulation tool shall implement a 1-tap (8.0 GT/s), a 2-tap (16.0 GT/s), a 3 tap DFE (32.0 GT/s), and a 16-tap DFE (64.0 GT/s), where the dynamic range for the feedback coefficient is defined in § Section 8.4.1.10 .
- Optimizing Tx equalization and Rx DFE/CTLE settings: The simulation tool shall implement an optimization algorithm that selects the combination of Tx equalization and Rx CTLE and DFE settings that yields a maximum value for the eye height (at the data sample point) multiplied by the eye width at the far end of the channel. For details refer to § Section 8.4.1.8 .
- Statistical Treatment of jitter: In order to avoid overestimating the effect of channel-data and channel-jitter interactions, the tool shall use a statistical analysis of these parameters to generate voltage/jitter eye margins.

8.5.1.3.3 Simulation Tool Outputs §

Output eye parameters: The simulator shall generate a statistically defined output that displays the eye width and eye height. EH will be measured as the peak eye height at the data sample location, while EW shall be measured at the zero-crossing line. Additionally the simulator shall have the capability to adjust the data sample point by ± 0.1 UI from the mean center of the UI for 8.0 and 16.0 GT/s as shown in § Figure 8-81 . For 32.0 GT/s the simulator shall adjust the sample point up to 0.30 UI to the left of the mean center of the UI sample position in 0.05 UI increments, computing the

DFE coefficients for each sample location and selecting the result producing the maximum value for the eye height (at the data sample point) multiplied by the eye width. For 64.0 GT/s the simulator shall adjust the sample point up to 0.30 UI to the left of the mean center of the UI sample position in 0.015 UI increments, computing the DFE coefficients for each sample location and selecting the result producing the maximum value for the eye height (at the data sample point) multiplied by the eye width.

8.5.1.3.4 Open Source Simulation Tool §

An open source simulation tool shall be provided with the specification as design collateral. The tool will provide a turnkey capability, where the user provides the channel characteristics at the Receiver's die pad as step responses, and the tool calculates a statistical eye showing pass/fail.

8.5.1.4 Behavioral Transmitter Parameters §

8.5.1.4.1 Deriving Voltage and Jitter Parameters §

This section is for informative purposes. The voltage and jitter parameters may be derived from the Transmitter voltage and jitter parameters but are referenced to the die pad. This is necessary to allow the channel simulation to include a behavioral Tx package and drive the package from the die pads. Additionally, the T_j terms must be decomposed into separate R_j and D_j terms.

- V_{TX-CH-FS-NO-EQ} and V_{TX-CH-RS-NO-EQ}: These two parameters define the minimum peak-peak voltage corresponding to V_d in § Figure 8-6.
- The jitter parameters are derived based on the following set of equations. Algebraic manipulation is used to extract the R_j implicitly defined by the combination of T_j and D_j terms. The following numbers are based on 8.0 GT/s Tx jitter parameters based on values specified in § Table 8-6. The same approach is used to extract jitter parameters for 2.5, 5.0, 16.0, 32.0, and 64.0 GT/s where it is assumed that maximum data rate supported by the PCIe 6.0 device is 64.0 GT/s and the jitter values in § Table 8-6 are used in the extraction process.

$$\text{jit_hfrj_nui} = (T_{\text{TX-UTJ}} - T_{\text{TX-UDJ-DD}})/14.06 = 1.11\text{ps}$$

$$T_{\text{TX-CH-UPW-RJ}} = (T_{\text{TX-UPWJ-TJ}} - T_{\text{TXUPWJ-DJDD}})/14.06 = 1.00\text{ps}$$

$$T_{\text{TX-CH-UPW-DJ}} = T_{\text{TXUPWJ-DJDD}} = 10.0\text{ps}$$

$$T_{\text{TX-CH-URJ}} = \sqrt{\text{jit_hfrj_nui}^2 - (T_{\text{TX-CH-UPW-RJ}} * 0.707)^2 + T_{\text{REFCLK-RMS}}^2} = 1.31\text{ps}$$

$$T_{\text{TX-CH-UDJDD}} = T_{\text{TX-UDJ-DD}} - (T_{\text{TXUPWJ-DJDD}})/2 = 7.00\text{ps}$$

A-0840A

Figure 8-80 Example Derivation of 8.0 GT/s Jitter Parameters for § Table 8-15 §

A channel must be tested at all data rates, with the corresponding Tx jitter parameters, that it is intended to support during normal operation. For example, a channel intended to support a maximum data rate of 8.0 GT/s must be tested at 2.5, 5.0, and 8.0 GT/s.

Table 8-15 Jitter/Voltage Parameters for Channel Tolerancing §

| Symbol | Parameter | Value | Units | Notes |
|-----------------------------|-----------------------|-------|-------|-----------------------|
| V _{TX-CH-FS-NO-EQ} | Full swing Tx voltage | 804 | mVPP | Full swing, No Tx Eq. |

| Symbol | Parameter | Value | Units | Notes |
|----------------------|--------------------------|-------|-------|--------------------------|
| $V_{TX-CH-RS-NO-EQ}$ | Reduced swing Tx voltage | 402 | mVPP | Reduced swing, No Tx Eq. |

2.5 GT/s Jitter Parameters and Voltage Parameters

| | | | | |
|-------------------------------|---|------|--------|--|
| $T_{TX-CH-URJ-2.5G}$ | Tx uncorrelated Rj | 3.45 | ps RMS | See Note 1 |
| $T_{TX-CH-UDJDD-2.5G}$ | Tx uncorrelated DjDD | 20 | ps PP | |
| $T_{TX-CH-UPW-RJ-2.5G}$ | Uncorrelated PW Rj | 1.42 | ps RMS | See Note 2 |
| $T_{TX-CH-UPW-DJ-2.5G}$ | PW DDj | 80 | ps PP | |
| $T_{TX-DIEPAD-EDGERATE-2.5G}$ | Signal edge rate at behavioral Tx die pad | 140 | ps | Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3. |

5.0 GT/s Jitter Parameters and Voltage Parameters

| | | | | |
|-----------------------------|---|------|--------|--|
| $T_{TX-CH-URJ-5G}$ | Tx uncorrelated Rj | 3.45 | ps RMS | See Note 1 |
| $T_{TX-CH-UDJDD-5G}$ | Tx uncorrelated DjDD | 20 - | ps PP | |
| $T_{TX-CH-UPW-RJ-5G}$ | Uncorrelated PW Rj | 1.42 | ps RMS | |
| $T_{TX-CH-UPW-DJ-5G}$ | PW DDj | 40 | ps PP | See Note 2. |
| $T_{TX-DIEPAD-EDGERATE-5G}$ | Signal edge rate at behavioral Tx die pad | 70 | ps | Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3. |

8.0 GT/s Jitter Parameters and Voltage Parameters

| | | | | |
|-----------------------------|---|-------|--------|--|
| $T_{TX-CH-URJ-8G}$ | Tx uncorrelated Rj | 1.31 | ps RMS | No DDj of HF jitter. See Note 1. |
| $T_{TX-CH-UDJDD-8G}$ | Tx uncorrelated DjDD | 7.0 | ps PP | No DDj of HF jitter |
| $T_{TX-CH-UPW-RJ-8G}$ | Uncorrelated PW Rj | 1.0 | ps RMS | |
| $T_{TX-CH-UPW-DJ-8G}$ | PW DDj | 10 | ps PP | See Note 2. |
| $T_{TX-DIEPAD-EDGERATE-8G}$ | Signal edge rate at behavioral Tx die pad | 43.75 | ps | Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3. |

16.0 GT/s Jitter Parameters and Voltage Parameters

| | | | | |
|------------------------------|---|--------|--------|--|
| $T_{TX-CH-URJ-16G}$ | Tx uncorrelated Rj | 0.71 | ps RMS | See Note 1. |
| $T_{TX-CH-UDJDD-16G}$ | Tx uncorrelated DjDD | 3.75 | ps PP | |
| $T_{TX-CH-UPW-RJ-16G}$ | Uncorrelated PW Rj | 0.54 | ps RMS | |
| $T_{TX-CH-UPW-DJ-16G}$ | PW DDj | 5.0 | ps PP | See Note 2. |
| $T_{TX-DIEPAD-EDGERATE-16G}$ | Signal edge rate at behavioral Tx die pad | 21.875 | ps | Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3. |

| Symbol | Parameter | Value | Units | Notes |
|---|---|-------|--------|--|
| 32.0 GT/s Jitter Parameters and Voltage Parameters | | | | |
| $T_{TX-CH-URJ-32G}$ | Tx uncorrelated Rj | 0.276 | ps RMS | See Note 1. |
| $T_{TX-CH-UDJDD-32G}$ | Tx uncorrelated DjDD | 1.875 | ps PP | |
| $T_{TX-CH-UPW-RJ-32G}$ | Uncorrelated PW Rj | 0.27 | ps RMS | |
| $T_{TX-CH-UPW-DJ-32G}$ | PW DDj | 2.5 | ps PP | See Note 2. |
| $T_{TX-DIEPAD-EDGERATE-32G}$ | Signal edge rate at behavioral Tx die pad | 10.94 | ps | Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3. |
| 64.0 GT/s Jitter Parameters and Voltage Parameters | | | | |
| $T_{TX-CH-URJ-64G}$ | Tx uncorrelated Rj | 0.215 | ps RMS | See Note 1. |
| $T_{TX-CH-UDJDD-64G}$ | Tx uncorrelated DjDD | 0.938 | ps PP | |
| $T_{TX-CH-UPW-RJ-64G}$ | Uncorrelated PW Rj | 0.289 | ps RMS | |
| $T_{TX-CH-UPW-DJ-64G}$ | PW DDj | 1.25 | ps PP | See Note 2. |
| $T_{TX-DIEPAD-EDGERATE-64G}$ | Signal edge rate at behavioral Tx die pad | 10.94 | ps | Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3. |

Notes:

1. Includes low frequency (non F/2) Rj components from the Transmitter and Rj from the Refclk.
2. Applied on a per edge basis as a dual Dirac model.
3. Does not include parasitic die pad capacitance. See § Figure 8-60 for details of behavioral package.

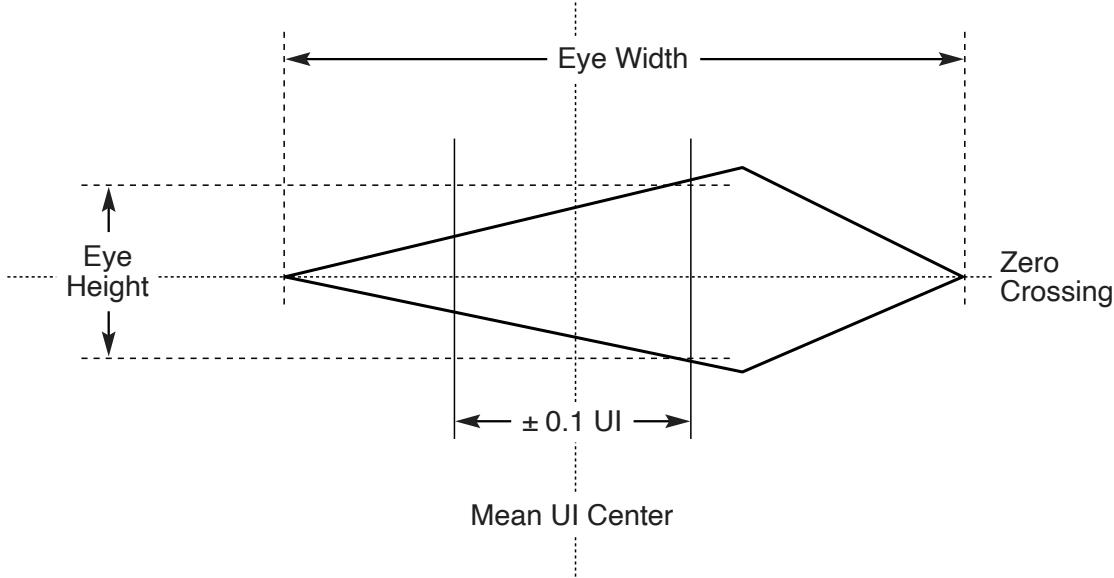
8.5.1.4.2 Optimizing Tx/Rx Equalization (8.0, 16.0, 32.0, and 64.0 GT/s only) §

The behavioral Receiver selects the combination of Transmitter Equalization, CTLE, DFE and sample location (32.0 and 64.0 GT/s only) that produces the optimal eye area (eye width multiplied by eye height).

8.5.1.4.3 Pass/Fail Eye Characteristics §

The output of the simulation tool shall be in the form of pass/fail characteristics as defined by an eye mask as shown in § Figure 8-81 . EH and EW must meet respectively the voltage and jitter parameters defined in § Table 8-16 . Eye margins are defined at the die pad of the Receiver after the appropriate Tx and Rx equalization algorithms have been applied. In the case where the channel is being designed for a specific pair of silicon devices and the package models of these silicon devices including crosstalk aggressors are known the actual device packages may be used instead of the reference packages in running the pad to pad channel pass/fail compliance simulations.

Base 6.4 vs Base 6.3



A-0841A

Figure 8-81 EH, EW Mask [§](#)

Note that the pass/fail EH and EW limits shown in § Figure 8-81 are identical to the limits defined for Rx testing in § Table 8-11 for 8.0 and 16.0 GT/s. For 32.0 and 64.0 GT/s, the pass/fail EH and EW limits are still identical but they are computed at the optimal sample location. For 2.5 and 5.0 GT/s the limits for Rx testing are referenced to the package pins and the limits for channel tolerancing in this table are referenced to the Receiver pad after applying the same reference package used for 8.0 GT/s channel tolerancing.

Table 8-16 Channel Tolerancing Eye Mask Values [§](#)

| Symbol | Parameter | Value | Units | Comments |
|-----------------------------|-------------------------------|-------------|-------|--|
| 2.5 GT/s Eye Margins | | | | |
| $V_{RX-CH-EH-2.5G}$ | Eye height | <130 (min) | mVPP | Eye height at BER=10 ⁻¹² . Note 1 |
| $T_{RX-CH-EW-2.5G}$ | Eye width at zero-crossing | <0.35 (min) | UI | Eye width at BER=10 ⁻¹² |
| $T_{RX-DS-OFFSET-2.5G}$ | Peak EH offset from UI center | ±0.1 | UI | See § Figure 8-81 for details. |
| 5.0 GT/s Eye Margins | | | | |
| $V_{RX-CH-EH-5G}$ | Eye height | < 85 (min) | mVPP | Eye height at BER=10 ⁻¹² . Note 1 |
| $T_{RX-CH-EW-5G}$ | Eye width at zero-crossing | <0.30 (min) | UI | Eye width at BER=10 ⁻¹² |
| $T_{RX-DS-OFFSET-5G}$ | Peak EH offset from UI center | ±0.1 | UI | See § Figure 8-81 for details. |
| 8.0 GT/s Eye Margins | | | | |

Base 6.4 vs Base 6.3

| Symbol | Parameter | Value | Units | Comments |
|-----------------------|---------------------------------|-----------|-------|--|
| $V_{RX-CH-EH-8G}$ | Eye height | 25 (min) | mVPP | Eye height at $BER=10^{-12}$. Note 1. |
| $T_{RX-CH-EW-8G}$ | Eye width at zero-crossing | 0.3 (min) | UI | Eye width at $BER=10^{-12}$ |
| $T_{RX-DS-OFFSET-8G}$ | Peak EH offset from UI center | ± 0.1 | UI | See § Figure 8-81 for details. |
| $V_{RX-DFE-D1-8G}$ | Range for DFE d_1 coefficient | ± 30 | mV | |

16.0 GT/s Eye Margins

| | | | | |
|------------------------|---------------------------------|-----------|------|--|
| $V_{RX-CH-EH-16G}$ | Eye height | 15 (min) | mVPP | Eye height at $BER=10^{-12}$. Note 1. |
| $T_{RX-CH-EW-16G}$ | Eye width at zero-crossing | 0.3 (min) | UI | Eye width at $BER=10^{-12}$ |
| $T_{RX-DS-OFFSET-16G}$ | Peak EH offset from UI center | ± 0.1 | UI | See § Figure 8-81 for details. |
| $V_{RX-DFE-D1-16G}$ | Range for DFE d_1 coefficient | ± 30 | mV | |
| $V_{RX-DFE-D2-16G}$ | Range for DFE d_2 coefficient | ± 20 | mV | |

32.0 GT/s Eye Margins

| | | | | |
|---------------------------------|---|--|------|--|
| $V_{RX-CH-EH-32G}$ | Eye height | 15 (min) | mVPP | Eye height at $BER=10^{-12}$. Note 1. |
| $T_{RX-CH-EW-32G}$ | Eye width at zero-crossing | 0.3 (min) | UI | Eye width at $BER=10^{-12}$ |
| $T_{RX-DS-OFFSET-32G}$ | Peak EH offset from UI center | N/A | UI | See § Figure 8-81 for details. |
| $T_{RX-SAMPLE-OFFSET-32G}$ | Max sample location offset to the left from UI center | 0.30 | UI | Note 2 |
| $T_{RX-SAMPLE-GRANULARITY-32G}$ | Granularity for sample location offset | 0.05 | UI | Note 2 |
| $V_{RX-DFE-D1-32G}$ | Range for DFE d_1 coefficient | $ d_1 / d_0$ (cursor amplitude) ≤ 0.8 | | |
| $V_{RX-DFE-D2-32G}$ | Range for DFE d_2 coefficient | ± 20 | mV | |
| $V_{RX-DFE-D3-32G}$ | Range for DFE d_3 coefficient | ± 20 | mV | |

64.0 GT/s Eye Margins

Base 6.4 vs Base 6.3

| Symbol | Parameter | Value | Units | Comments |
|------------------------------------|--|---|-------|--|
| $V_{RX-CH-TOP-EH-64G}$ | Top Eye height | 6 (min) | mVPP | Eye height at BER=10 ⁻⁶ . Note 1. |
| $T_{RX-CH-TOP-EW-64G}$ | Top Eye width at zero-crossing | 0.1 (min) | UI | Eye width at BER=10 ⁻⁶ |
| $T_{RX-DS-OFFSET-64G}$ | Peak EH offset from UI center | N/A | UI | |
| $T_{RX-SAMPLE-OFFSET-64G}$ | Max sample location offset to the left from UI center | 0.30 | UI | Note 2 |
| $T_{RX-SAMPLE-GRANULARITY-64G}$ | Granularity for sample location offset | 0.015 | UI | Note 2 |
| $V_{RX-DFE-D1-64G}$ | Range for DFE d1 coefficient | $ d_1 / d_0$ (cursor amplitude) < 0.55 | | |
| $V_{RX-DFE-TAPS-WEIGHTED-SUM-64G}$ | Range for weighted sum of absolute values of DFE d1-d16 coefficients | $(d_1 + d_2 + 0.85 \times d_3 + 0.6 \times d_4 + 0.25 \times d_5 + 0.1 \times d_6 + 0.05 \times \{ d_7 + d_8 + d_9 + d_{10} + d_{11} + d_{12} + d_{13} + d_{14} + d_{15} + d_{16} \}) / d_0 < 0.85$ | mV | |

Notes:

1. $V_{RX-CH-EH}$ is defined as max EH within an aperture of ± 0.1 UI from mean UI center.
2. The optimal eye area is computed at each offset from mean UI center -0.30 UI to mean UI center with the specified granularity.

8.5.1.4.4 Characterizing Channel Common Mode Noise §

A channel must meet the common mode requirements as they are defined in the Receiver specification. In general, it is not possible to accurately simulate all the channel's common mode noise contributions due to the large number of mechanisms that can generate CM noise, including the Transmitter. Typically channel common mode noise is a budgeted parameter, and the limits defined below assume a budgeting process. The channel's CM limit is defined as the amount of CM noise that a channel can add and still meet the Rx CM limits assuming the worst-case Tx CM.

Note that the Tx and channel CM noise parameters cannot simply be added to obtain the Rx CM limit. This is due to the fact that a channel will attenuate some of high frequency Tx CM noise while propagating Tx LF CM noise through with little loss. The channel may also contribute both high and low frequency CM components of its own.

8.5.1.4.5 Verifying $V_{CH-IDLE-DET-DIFF-pp}$ §

$V_{CH-IDLE-DET-DIFF-pp}$ is defined to guarantee that, when a Transmitter issues an EIEOS sequence, the Receiver is guaranteed to detect it. Potentially larger Transmitter equalization boost ratios at 8.0, 16.0, 32.0, and 64.0 GT/s necessitate that this parameter be verified; this procedure was not necessary for 2.5 or 5.0 GT/s, where the max Transmitter equalization boost is smaller. Defining the launch and detect voltages at the Tx/Rx die pad permits $V_{CH-IDLE-DET-DIFF-pp}$ to be verified with the same channel and Tx/Rx package models used to determine eye margins. It is also acceptable to simulate from Tx pin to Rx pin (excluding the Tx and Rx behavioral package models), in which case the EIEOS and idle detect parameters defined in the Tx and Rx sections are applicable.

Long channels, where $V_{TX-EIEOS-FS}$ is applicable, are characterized by driving the channel under test with the EIEOS pattern and -11.0 dB de-emphasis and zero dB preshoot. For short channels, where $V_{TX-EIEOS-RS}$ is applicable, -4.5 dB of de-emphasis and zero dB of preshoot are applied.

Table 8-17 EIEOS Signaling Parameters §

| Parameter | Description | Value | Units | Comments |
|-----------------------|--|-------|-------|---|
| $V_{CH-IDLE-EXIT-pp}$ | Idle detect voltage seen at the Rx die pad | 172 | mVPP | Assuming Rx RTERM of $2 \times 50 \Omega$ |
| $V_{CH-EIEOS-FS-Vb}$ | PP voltage during Vb interval at behavioral Tx die pad for full swing signaling | 255 | mVPP | Assuming Tx RS of $2 \times 50 \Omega$ |
| $V_{CH-EIEOS-RS-Vb}$ | PP voltage during Vb interval at behavioral Tx die pad for reduced swing signaling | 237 | mVPP | Assuming Tx RS of $2 \times 50 \Omega$ |

8.6 Refclk Specifications §

This version of the specification consolidates and streamlines the Refclk requirements. The 2.5 GT/s Refclk parameters are moved from the CEM spec to this spec so that all Refclk parameters for all data rates (2.5, 5.0, 8.0, 16.0, 32.0, and 64.0 GT/s) are now contained in this section.

8.6.1 Refclk Test Setup §

The test setup for the Refclk assumes that only the Refclk generator itself is present. Provision is made in the test setup to account for signal degradation that occurs between the pins of the Refclk generator and the Transmitter or Receiver in an actual system. The above described setup emulates the worst case signal degradation that is likely to occur at the pins of a PCI Express device. Note that the Refclk signal is tested into a load that represents the series (open) termination appearing at the Refclk input pins of a PCIe device for all requirements except 32.0 and 64.0 GT/s reference clock jitter. For 32.0 and 64.0 GT/s, the reference clock jitter is measured with an oscilloscope, and is tested with the reference clock terminated by 50 Ohm terminations without a channel.

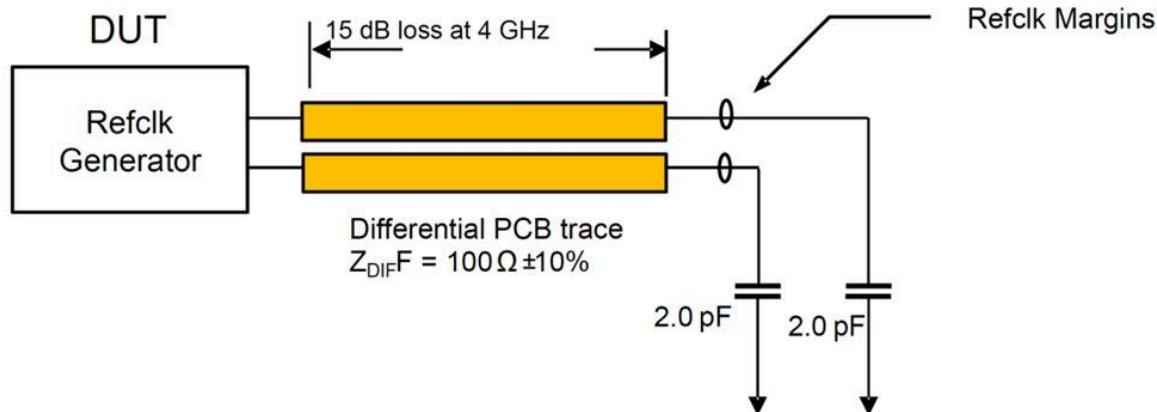


Figure 8-82 Oscilloscope Refclk Test Setup for All Cases Except Jitter at 32.0 and 64.0 GT/s §

Base 6.4 vs Base 6.3

8.6.2 REFCLK AC Specifications §

All specifications in § Table 8-18 are to be measured using a test configuration as described in Note 11 with a circuit as shown in § Figure 8-82 .

Table 8-18 REFCLK DC Specifications and AC Timing Requirements §

| Symbol | Parameter | 100 MHz Input | | Unit | Note |
|---|---|---------------|-------|------|-----------|
| | | Min | Max | | |
| Rising Edge Rate | Rising Edge Rate | 0.6 | 4.0 | V/ns | 2, 3 |
| Falling Edge Rate | Falling Edge Rate | 0.6 | 4.0 | V/ns | 2, 3 |
| V_{IH} | Differential Input High Voltage | +150 | | mV | 2 |
| V_{IL} | Differential Input Low Voltage | | -150 | mV | 2 |
| V_{CROSS} | Absolute crossing point voltage | +250 | +550 | mV | 1, 4, 5 |
| $V_{CROSS\ DELTA}$ | Variation of V_{CROSS} over all rising clock edges | | +140 | mV | 1, 4, 9 |
| V_{RB} | Ring-back Voltage Margin | -100 | +100 | mV | 2, 12 |
| T_{STABLE} | Time before V_{RB} is allowed | 500 | | ps | 2, 12 |
| $T_{PERIOD\ AVG}$ | Average Clock Period Accuracy | -300 | +2800 | ppm | 2, 10, 13 |
| $T_{PERIOD\ AVG_32G_64G_CC}$ | Average Clock Period Accuracy for devices that support 32.0 and 64.0 GT/s in CC Mode at any speed | -100 | +2600 | ppm | 2, 10, 13 |

| Symbol | Parameter | 100 MHz Input | | Unit | Note |
|---------------------------------------|---|---------------|--------|----------|-----------------|
| | | Min | Max | | |
| T_{PERIOD} $AVG_32G_64G_SRIS$ | Average Clock Period Accuracy for devices that support 32.0 and 64.0 GT/s in SRIS Mode at any speed | -100 | +1600 | ppm | 2, 10, 13 |
| $T_{PERIOD\ ABS}$ | Absolute Period (including Jitter and Spread Spectrum modulation) | 9.847 | 10.203 | ns | 2, 6 |
| T_{PERIOD} $ABS_32G_64G_CC$ | Absolute Period (including Jitter and Spread Spectrum modulation) for devices that support 32.0 and 64.0 GT/s in CC Mode at any speed | 9.849 | 10.201 | ns | 2, 6 |
| T_{PERIOD} $ABS_32G_64G_SRIS$ | Absolute Period (including Jitter and Spread Spectrum modulation) for devices that support 32.0 and 64.0 GT/s in SRIS Mode at any speed | 9.849 | 10.181 | ns | 2, 6 |
| $T_{CCJITTER}$ | Cycle to Cycle jitter | | 150 | ps | 2 |
| V_{MAX} | Absolute Max input voltage | | +1.15 | V | 1, 7 |
| V_{MIN} | Absolute Min input voltage | | -0.3 | V | 1, 8 |
| Duty Cycle | Duty Cycle | 40 | 60 | % | 2 |
| Rise-Fall Matching | Rising edge rate (REFCLK+) to falling edge rate (REFCLK-) matching | | 20 | % | 1, 14 |
| Z_{C-DC} | Clock source DC impedance | 40 | 60 | Ω | 1, 11 |

Notes:

1. Measurement taken from single ended waveform.
2. Measurement taken from differential waveform.
3. Measured from -150 mV to +150 mV on the differential waveform (derived from REFCLK+ minus REFCLK-). The signal must be monotonic through the measurement region for rise and fall time. The 300 mV measurement window is centered on the differential zero-crossing. See § Figure 8-87 .
4. Measured at crossing point where the instantaneous voltage value of the rising edge of REFCLK+ equals the falling edge of REFCLK-. See § Figure 8-83 .
5. Refers to the total variation from the lowest crossing point to the highest, regardless of which edge is crossing. Refers to all crossing points for this measurement. See § Figure 8-83 .
6. Defines as the absolute minimum or maximum instantaneous period. This includes cycle to cycle jitter, relative PPM tolerance, and spread spectrum modulation. See § Figure 8-86 .
7. Defined as the maximum instantaneous voltage including overshoot. See § Figure 8-83 .
8. Defined as the minimum instantaneous voltage including undershoot. See § Figure 8-83 .
9. Defined as the total variation of all crossing voltages of Rising REFCLK+ and Falling REFCLK-. This is the maximum allowed variance in V_{CROSS} for any system. See § Figure 8-84 .
10. Note deleted.
11. REFCLK+ and REFCLK- are to be measured at the load capacitors C_L . Single ended probes must be used for measurements requiring single ended measurements. Either single ended probes with math or differential probe can be used for differential measurements. Test load $C_L = 2 \text{ pF}$.
12. T_{STABLE} is the time the differential clock must maintain a minimum $\pm 150 \text{ mV}$ differential voltage after rising/falling edges before it is allowed to droop back into the $V_{RB} \pm 100 \text{ mV}$ differential range. See § Figure 8-88 .

| Symbol | Parameter | 100 MHz Input | | Unit | Note |
|--------|---|---------------|-----|------|------|
| | | Min | Max | | |
| 13. | PPM refers to parts per million and is a DC absolute period accuracy specification. 1 PPM is $1/1,000,000^{\text{th}}$ of 100.000000 MHz exactly or 100 Hz. For example, for 300 PPM, then we have an error budget of 100 Hz/PPM \times 300 PPM = 30 kHz. The period is to be measured with a frequency counter with measurement window set to 100 ms or greater. | | | | |
| 14. | Matching applies to rising edge rate for REFCLK+ and falling edge rate for REFCLK-. It is measured using a ± 75 mV window centered on the median cross point where REFCLK+ rising meets REFCLK- falling. The median cross point is used to calculate the voltage thresholds the oscilloscope is to use for the edge rate calculations. The Rise Edge Rate of REFCLK+ should be compared to the Fall Edge Rate of REFCLK-; the maximum allowed difference should not exceed 20% of the slowest edge rate. See § Figure 8-85. | | | | |

Base 6.4 vs Base 6.3

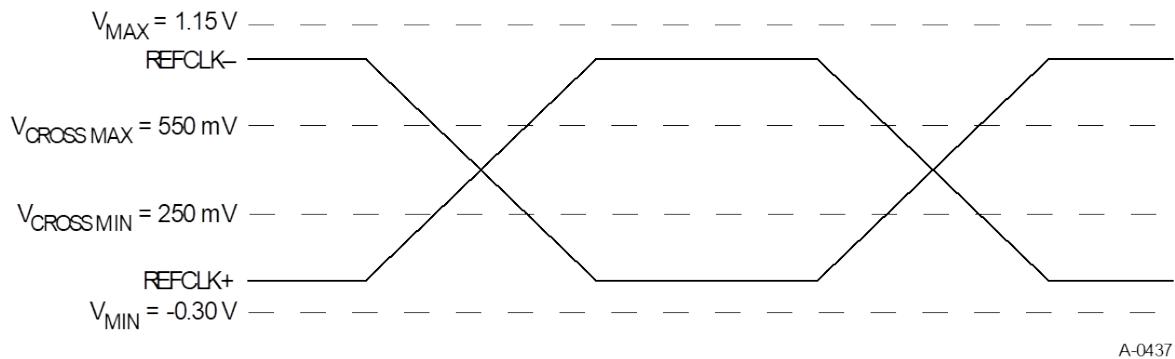


Figure 8-83 Single-Ended Measurement Points for Absolute Cross Point and Swing §

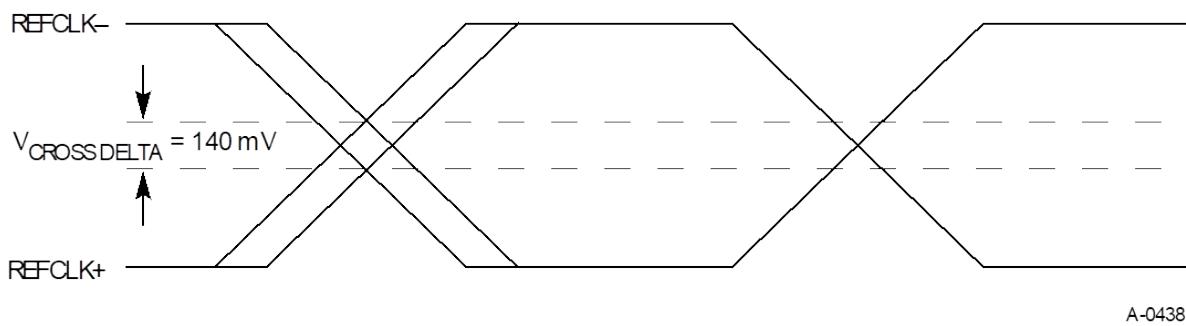


Figure 8-84 Single-Ended Measurement Points for Delta Cross Point §

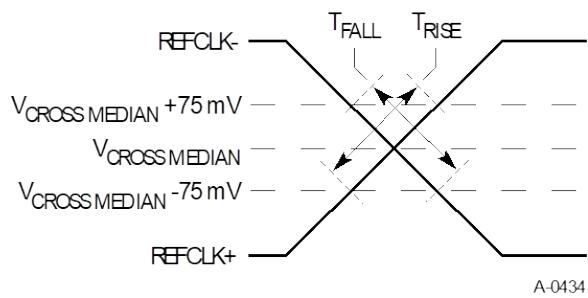
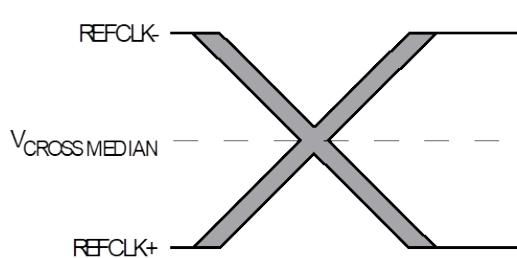


Figure 8-85 Single-Ended Measurement Points for Rise and Fall Time Matching §

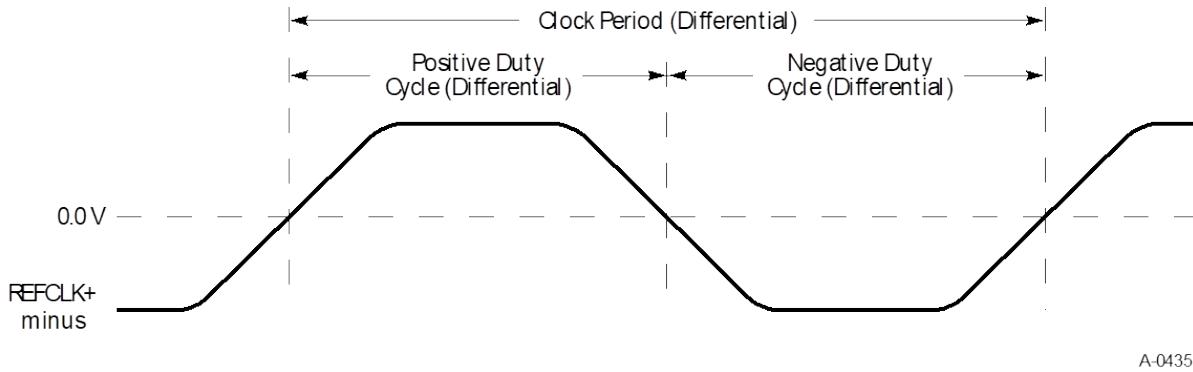


Figure 8-86 Differential Measurement Points for Duty Cycle and Period §

Base 6.4 vs Base 6.3

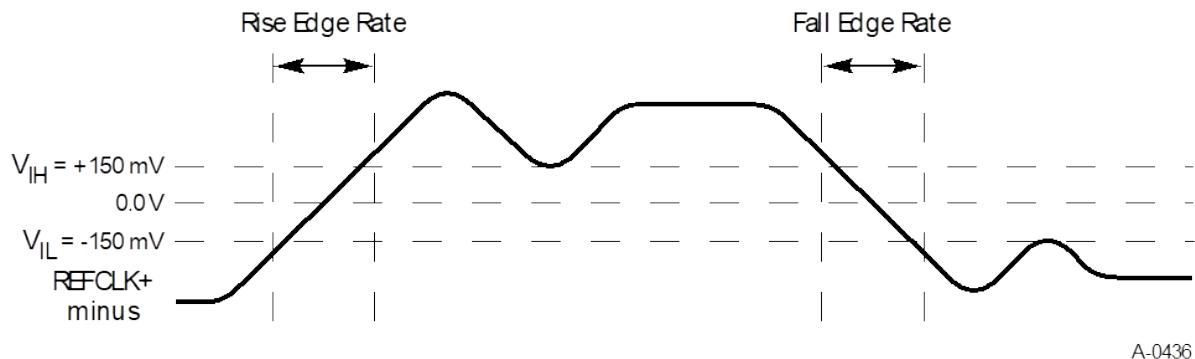


Figure 8-87 Differential Measurement Points for Rise and Fall Time §

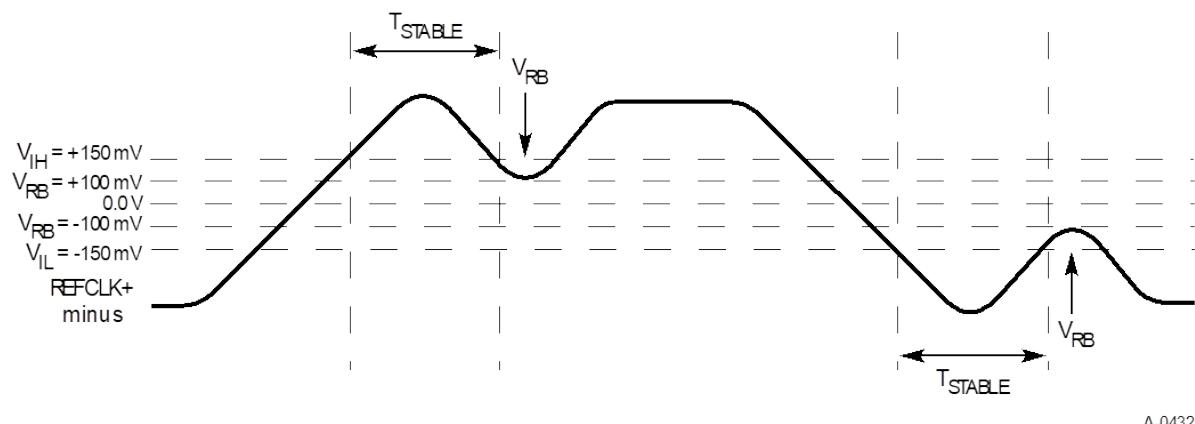


Figure 8-88 Differential Measurement Points for Ringback §

8.6.3 Data Rate Independent Refclk Parameters §

A number of Refclk parameters are data rate independent and are listed in the table below. $T_{TRANSPORT-DELAY}$ is defined in § Section 8.6.6 and illustrated in § Figure 8-91 . It is relevant only for the Common Refclk architecture. For the SRIS mode the source of the SSC modulation is implementation dependent.

Table 8-19 Data Rate Independent Refclk Parameters §

| Symbol | Description | Limits | Units | Notes |
|--|--|-----------------------------|------------|-------|
| F_{REFCLK} | Refclk Frequency | 99.97 (min) 100.03 (max) | MHz | |
| $F_{REFCLK_32G_64G}$ | Refclk Frequency for devices that support 32.0 and 64.0 GT/s | 99.99 (min) 100.01 (max) | MHz | |
| F_{SSC} | SSC frequency range | 30 (min) 33 (max) | kHz | 3 |
| $T_{SSC-FREQ-DEVIATION}$ | SSC deviation | -0.5 (min) 0.0 (max) | % | 3 |
| $T_{SSC-FREQ-DEVIATION_32G_64G_SRIS}$ | SSC deviation for devices that support 32.0 and 64.0 GT/s and SRIS when operating in SRIS mode at all speeds | -0.3 (min) 0.0 (max) | % | 3 |
| $T_{TRANSPORT-DELAY}$ | Tx-Rx transport delay | 12 (max) | ns | 1, 4 |
| $T_{SSC-MAX-FREQ-SLEW}$ | Max SSC df/dt | 1250 | ppm/ μs | 2, 3 |

Notes:

1. Parameter is relevant only for Common Refclk architecture.
 2. Measurement is made over 0.5 μs time interval with a 1st order LPF with an f_c of 60x the modulation frequency.
 3. When testing the a device configured for the IR reference clock architecture the SSC related parameters must be tested with the Tx output data instead of the reference clock.
 4. There are form factors (for example topologies including long cables) that may exceed the transport delay limit. Extra jitter from the large transport delay must be accounted by these form factor specifications.
-

8.6.3.1 Low Frequency Refclk Jitter Limits §

Refclks supporting SSC must meet an additional jitter limit over a range of low frequencies. Low frequency Refclk jitter limits are defined as a continuous, piece-wise linear graph from 30 kHz to 500 kHz as shown below. Unfiltered Refclk phase jitter must fall below this graph over the frequency range of interest.

Base 6.4 vs Base 6.3

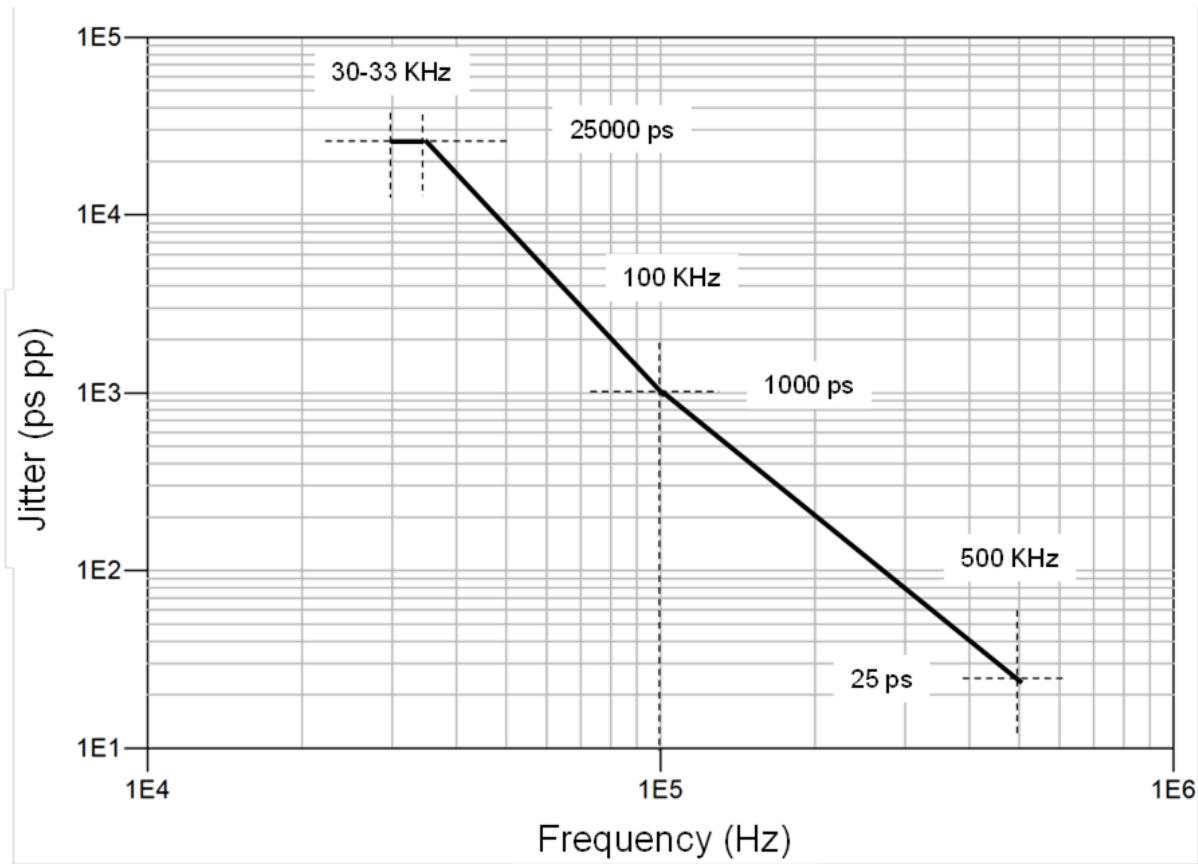


Figure 8-89 Limits for phase jitter from the Reference with 5000 ppm SSC §

8.6.4 Refclk Architectures Supported §

Two Refclk architectures are supported: **Common Refclk (CC)** and **Independent Refclk (IR)**. The CC clock architecture is described in terms of its topology and its corresponding jitter transfer function based on PLL and CDR characteristics. Finally, the corresponding Refclk jitter limits are given for each data rate. The jitter transfer function and corresponding jitter limits are not defined for the IR clock architecture. It is up to the implementer to trade off reference clock jitter and PLL characteristics to ensure that Transmitter requirements are met in the IR clocking mode.

8.6.5 Filtering Functions Applied to Raw Data §

Two types of filtering are applied to the raw Refclk data. The first, edge filtering, minimizes the measurement-induced jitter due to the finite sampling rate of the test equipment. The second, replicates the jitter filtering that is inherent in the combination of Tx/Rx PLLs, the Rx CDR and (where applicable) the transport delay. The combination of the preceding filter functions yields the effective Refclk jitter as it appears at the sample latch of the Receiver.

Note that the PLL and CDR filter functions represent minimally capable approximations to actual Receiver implementations and are not intended to define actual PLL or CDR implementations.

8.6.5.1 PLL Filter Transfer Function Example §

All PLLs are behaviorally modeled with a second order transfer function, $H(s)$, as defined in § Figure 8-91. The parameters defining the transfer function include the damping factor ζ and the natural frequency ω_n .

The relation between the 2nd order PLL (lowpass) natural frequency, ω_n and the 3 dB point ω_{3dB} , is given by the following expression:

$$\frac{\omega_{3dB}}{\omega_n} = \sqrt{\sqrt{1/(2\zeta^2 + 1)^2 + 1} + 2\zeta^2 + 1}$$

Equation 8-28 Relationship between 2nd order PLL natural frequency and 3 dB point §

The following plot of a 2nd order PLL illustrates the transfer function with an f_{3dB} of 5.0 MHz and 1.0 dB of peaking. This corresponds to $\zeta = 1.15$, and $\omega_n = 11.55$ Mrad/sec.

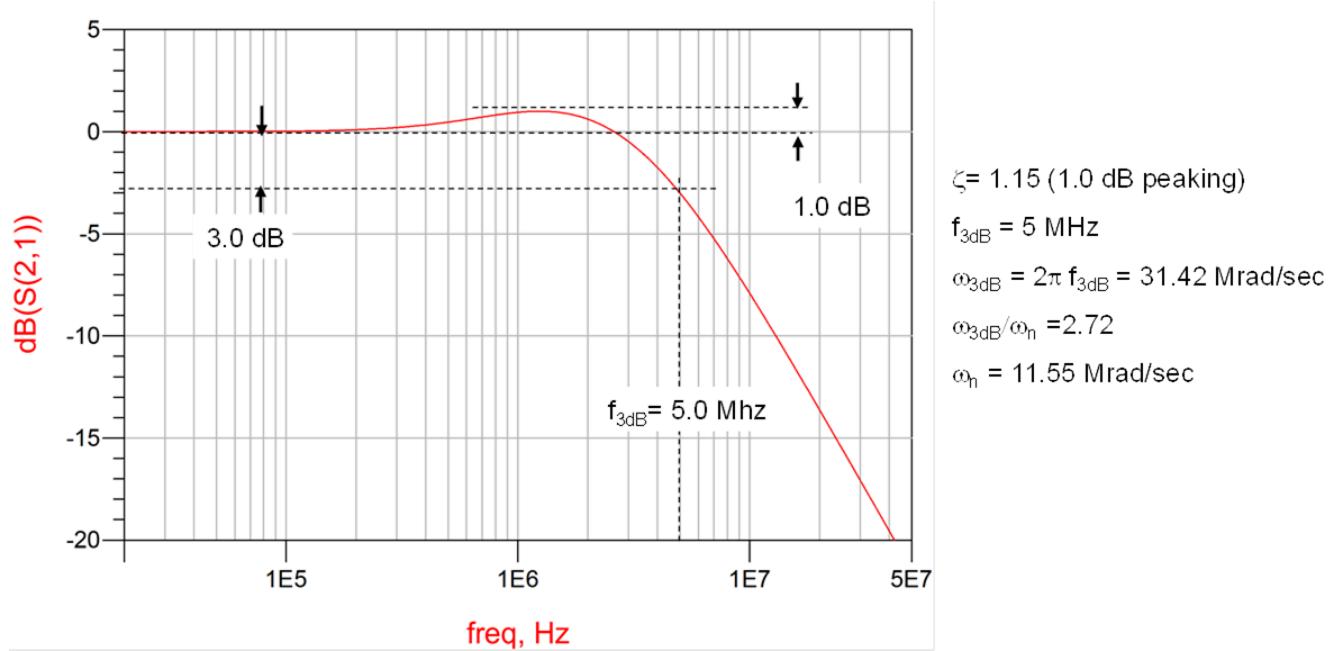


Figure 8-90 5 MHz PLL Transfer Function Example §

8.6.5.2 CDR Transfer Function Examples §

Depending on the Refclk architecture and data rate, either a first or higher order transfer function shall be used as a behavioral CDR bounding limit.

For behavioral CDR functions refer to § Section 8.3.5.5

8.6.6 Common Refclk Rx Architecture (CC) §

This architecture utilizes a single Refclk source that is distributed to both the Tx and Rx. Most of the SSC jitter sourced by the Refclk is propagated equally through Tx and Rx PLLs, and so intrinsically tracks LF jitter. This is particularly true for SSC which tends to be low frequency. § Figure 8-91 illustrates the Common Refclk Rx architecture, showing key jitter, delay, and PLL and CDR transfer function sources for all data rates except 32.0 and 64.0 GT/s. At 32.0 and 64.0 GT/s the only difference in the figure is Behavioral CDR transfer function as defined in § Section 8.3.5.5 . The amount of jitter appearing at the CDR is then defined by the difference function between the Tx and Rx PLLs multiplied by the CDR highpass characteristic.

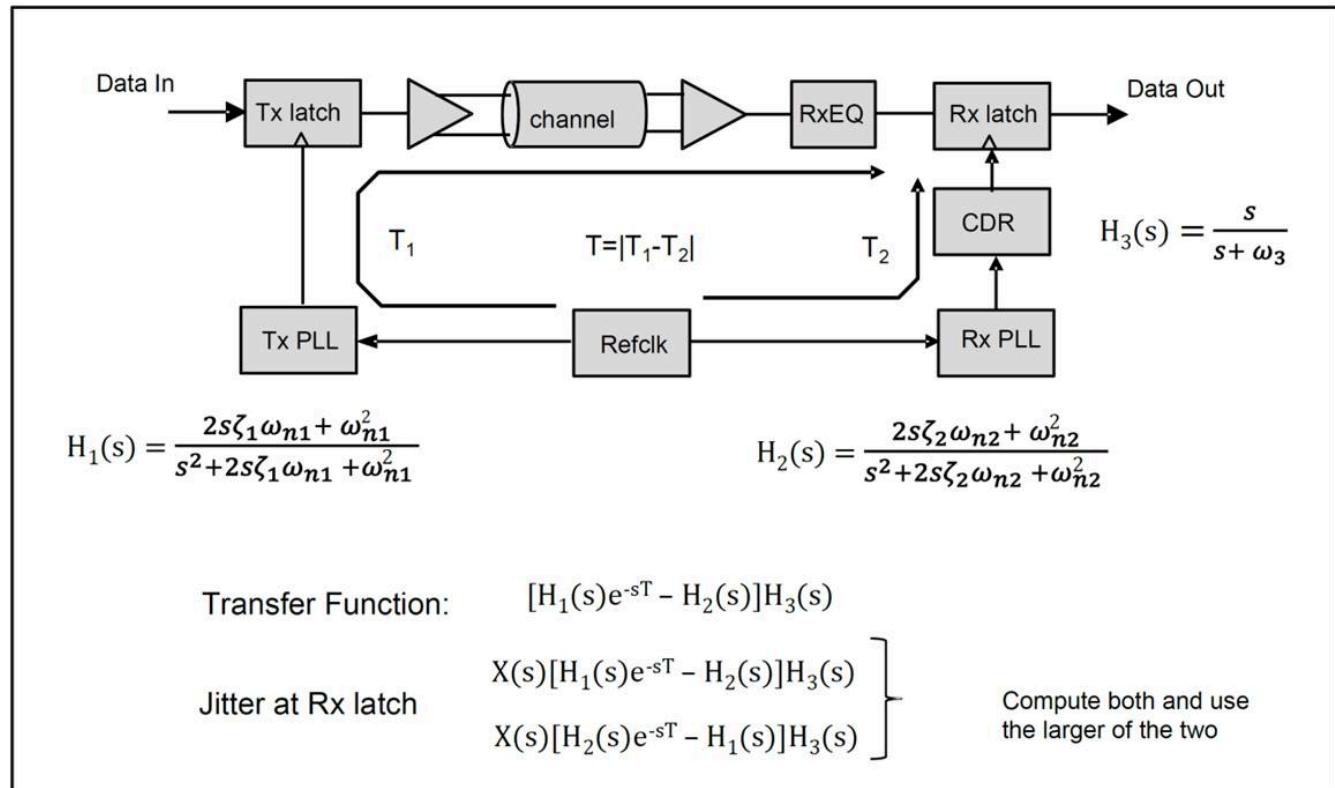


Figure 8-91 Common Refclk Rx Architecture for all Data Rates Except 32.0 and 64.0 GT/s §

Based on the above clock architecture, it is possible to define a difference function that corresponds to the worst case mismatch between Tx and Rx PLLs. Second order PLL transfer functions are assumed, (even though most PLL transfer functions are 3rd order or higher), since a 2nd order function tends to yield a slightly conservative difference function vis-a-vis most actual PLL implementations.

In the Common Refclk Rx architecture it is also necessary to comprehend a maximum Transmitter to Receiver transport delay difference. This delay delta is illustrated in § Figure 8-91 and represents the delay difference between the Transmitter data and recovered Receiver clock as seen at the inputs to the receiver's data latch.

8.6.6.1 Determining the Number of PLL BW and peaking Combinations §

A Tx or Rx PLL is defined by the combination of min/max bandwidth and peaking, making for a total of four possible limits. In the CC architecture both the Tx and Rx PLLs contribute to the jitter transfer function. At 2.5 GT/s only one set of BW/peaking limits is defined. If the combinations for the Tx and Rx PLLs limits are defined by the sets ($A_{TX}, B_{TX}, C_{TX}, D_{TX}$), ($A_{RX}, B_{RX}, C_{RX}, D_{RX}$) then it's easily demonstrated that there are a total of ten ways of selecting one element from each set. The delay term e^{-sT} can be applied to either $H_1(s)$ or $H_2(s)$, adding another six possibilities. Only six, as opposed to ten, terms are added when the delay term is considered because terms like A_{TX}, A_{RX} , are identical to A_{RX}, A_{TX} .

At 5.0 GT/s and higher data rates, two possible sets of limits of PLL bandwidth and peaking are defined, which may be defined by the sets ($A_{TX}, B_{TX}, C_{TX}, D_{TX}$) and ($E_{RX}, F_{RX}, G_{RX}, H_{RX}$). In this case the number of unique 2-element combinations from the above 4-element sets is 36, which increases to 64 when the delay term is considered.

8.6.6.2 CDR and PLL BW and Peaking Limits for Common Refclk §

The Common Refclk architecture filter function is dependent on the difference function defined by the BW and peaking of Tx and the Rx PLLs, plus the CDR high-pass characteristic. It is necessary to consider all corner case combinations of Tx and Rx PLL peaking and bandwidth plus the CDR characteristics at their minimum and maximum peaking values.

This procedure must be applied to all six data rates.

§ Figure 8-92 lists the PLL BW and CDR BW/peaking values that need to be applied as filter functions for 2.5 GT/s data rates. It is necessary to assign a min and a max value for peaking for the 2.5 GT/s PLL. The values of 0.01 dB and 3.0 dB represent best estimates of realistic PLL implementations.

The minimum peaking for 2.5 and 5.0 GT/s has been reduced to 0.01 dB to bring it into line with the 8.0 and 16.0 GT/s cases. Note that the Rx CDR is 1st order, so its natural frequency, ω_n can be directly obtained from its BW, unlike the 2nd order CDRs, where ω_n is a function of both BW and peaking. For 32.0 GT/s, a 2nd-order CDR filter with 20 MHz BW described by § Equation 8-25 must be used.

| PLL #1, PLL #2 | | 0.01 dB peaking | 3.0 dB peaking | BW _{CDR} (min) = 1.5 MHz, 1 st order | CDR |
|-----------------------------------|--|---|--|--|-----|
| BW _{PLL} (min) = 1.5 MHz | | $\omega_{n1} = .336 \text{ Mrad/s}$ $\zeta_1 = 14$ | $\omega_{n1} = 5.09 \text{ Mrad/s}$ $\zeta_1 = 0.54$ | | |
| BW _{PLL} (max) = 22 MHz | | $\omega_{n1} = 4.93 \text{ Mrad/s}$ $\zeta_1 = 14$ | $\omega_{n1} = 74.68 \text{ Mrad/s}$ $\zeta_1 = 0.54$ | | |

16 combinations 2.5 GT/s

Figure 8-92 Common Refclk PLL and CDR Characteristics for 2.5 GT/s §

PLL and CDR jitter and peaking characteristics for 5.0, 8.0, and 16.0 GT/s yield a larger number of possible combinations because two sets of PLL BW and peaking limits are given. This choice to support two sets of BW and peaking was made to give designers as much latitude as possible when designing PLL circuits.

Base 6.4 vs Base 6.3

| | | | | | |
|---|---|--|----------------------------|---|--|
| PLL #1 | 0.01 dB peaking | 1.0 dB peaking | PLL #2 | 0.01 dB peaking | 3.0 dB peaking |
| $BW_{PLL}(\min) = 5.0$ MHz | $\omega_{n1} = 1.12$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 11.01$ Mrad/s $\zeta_1 = 1.16$ | $BW_{PLL}(\min) = 8.0$ MHz | $\omega_{n2} = 1.79$ Mrad/s $\zeta_2 = 14$ | $\omega_{n2} = 26.86$ Mrad/s $\zeta_2 = 0.54$ |
| $BW_{PLL}(\max) = 16$ MHz | $\omega_{n1} = 3.58$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 35.26$ Mrad/s $\zeta_1 = 1.16$ | $BW_{PLL}(\max) = 16$ MHz | $\omega_{n2} = 3.58$ Mrad/s $\zeta_2 = 14$ | $\omega_{n2} = 53.73$ Mrad/s $\zeta_2 = 0.54$ |
| $BW_{CDR}(\min) = 5$ MHz, 1 st order | | | | | 64 combinations |
| | | | | | 5 GT/s |

Figure 8-93 Common Refclk PLL and CDR Characteristics for 5.0 GT/s §

| | | | | | |
|--|--|--|----------------------------|--|--|
| PLL #1 | 0.01 dB peaking | 2.0 dB peaking | PLL #2 | 0.01 dB peaking | 1.0 dB peaking |
| $BW_{PLL}(\min) = 2.0$ MHz | $\omega_{n1} = 0.448$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 6.02$ Mrad/s $\zeta_1 = 0.73$ | $BW_{PLL}(\min) = 2.0$ MHz | $\omega_{n2} = 0.448$ Mrad/s $\zeta_2 = 14$ | $\omega_{n2} = 4.62$ Mrad/s $\zeta_2 = 1.15$ |
| $BW_{PLL}(\max) = 4.0$ MHz | $\omega_{n1} = 0.896$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 12.04$ Mrad/s $\zeta_1 = 0.73$ | $BW_{PLL}(\max) = 5.0$ MHz | $\omega_{n2} = 1.12$ Mrad/s $\zeta_2 = 14$ | $\omega_{n2} = 11.53$ Mrad/s $\zeta_2 = 1.15$ |
| $BW_{CDR}(\min) = 10$ MHz, 1 st order | | | | | 64 combinations |
| | | | | | 8.0, 16.0 GT/s |

Figure 8-94 Common Refclk PLL and CDR Characteristics for 8.0 and 16.0 GT/s §

| | | | | |
|----------------------------|--|---|-----------------|-----------|
| PLL #1, PLL #2 | 0.01 dB peaking | 2.0 dB peaking | 32.0 GT/s CC | CDR |
| $BW_{PLL}(\min) = 0.5$ MHz | $\omega_{n1} = 0.112$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 1.50$ Mrad/s $\zeta_1 = 0.73$ | | |
| $BW_{PLL}(\max) = 1.8$ MHz | $\omega_{n1} = 0.403$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 5.42$ Mrad/s $\zeta_1 = 0.73$ | 16 combinations | |
| | | | | 32.0 GT/s |

Figure 8-95 Common Refclk PLL and CDR Characteristics for 32.0 GT/s §

| | | | | |
|----------------------------|--|---|-----------------|-----------|
| PLL #1, PLL #2 | 0.01 dB peaking | 2.0 dB peaking | 64.0 GT/s CC | CDR |
| $BW_{PLL}(\min) = 0.5$ MHz | $\omega_{n1} = 0.112$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 1.50$ Mrad/s $\zeta_1 = 0.73$ | | |
| $BW_{PLL}(\max) = 1.0$ MHz | $\omega_{n1} = 0.224$ Mrad/s $\zeta_1 = 14$ | $\omega_{n1} = 3.00$ Mrad/s $\zeta_1 = 0.73$ | 16 combinations | |
| | | | | 64.0 GT/s |

Figure 8-96 Common Refclk PLL and CDR Characteristics for 64.0 GT/s §

8.6.7 Jitter Limits for Refclk Architectures §

§ Table 8-20 lists the jitter limits for the CC Refclk architecture at each of the four data rates.

Jitter at 2.5 GT/s is measured as a peak to peak jitter value, because a substantial proportion of the jitter is SSC harmonics which appears at the receiver as Dj. The combination of the 2.5 GT/s PLL and CDR bandwidths passes a significant amount of SSC residual, where it appears Dj.

For 5.0, 8.0, and 16.0 GT/s jitter is specified as an RMS (Rj) value. These signaling speeds utilize a lower PLL BW and a higher CDR BW, and the effect is to suppress SSC harmonics such that almost all the jitter appears as Rj.

Table 8-20 Jitter Limits for CC Architecture §

| Data Rate | CC jitter Limit | Notes |
|-----------|-----------------|------------|
| 2.5 GT/s | 86 ps pp | 1, 2 |
| 5.0 GT/s | 3.1 ps RMS | 1, 2 |
| 8.0 GT/s | 1.0 ps RMS | 1, 2 |
| 16.0 GT/s | 0.5 ps RMS | 1, 2, 3, 4 |
| 32.0 GT/s | 0.15 ps RMS | 1, 2, 3, 5 |
| 64.0 GT/s | 0.1 ps RMS | 1, 2, 3, 6 |

Notes:

1. The Refclk jitter is measured after applying the filter function in § Figure 8-91
2. Jitter measurements shall be made with a capture of at least 100,000 clock cycles captured by a real time oscilloscope (RTO) with a sample rate of 20 GS/s or greater. Broadband oscilloscope noise must be minimized in the measurement. The measured PP jitter is used (no extrapolation) for RTO measurements. Alternately - Jitter measurements may be used with a Phase Noise Analyzer (PNA) extending (flat) and integrating and folding the frequency content up to an offset from the carrier frequency of at least 200 MHz (at 300 MHz absolute frequency) below the Nyquist frequency. For PNA measurements for the 2.5 GT/s data rate the RMS jitter is converted to peak to peak jitter using a multiplication factor of 8.83. In the case where real time oscilloscope and PNA measurements have both been done and produce different results the RTO result must be used.
3. For the 16.0, 32.0, and 64.0 GT/s CC measurements SSC spurs from the fundamental and harmonics are removed up to a cutoff frequency of 2 MHz taking care to minimize removal of any non-SSC content.
4. Note that 0.7 ps RMS is to be used in channel simulations to account for additional noise in a real system.
5. Note that 0.25 ps RMS is to be used in channel simulations to account for additional noise in a real system.
6. Note that 0.15 ps RMS is to be used in channel simulations to account for additional noise in a real system.

8.6.8 Form Factor Requirements for RefClock Architectures §

Each form factor specification must include the following table (see § Table 8-21) to provide a clear summary of the clocking architecture requirements for devices that support the form factor specification. For each clocking architecture the table indicates whether that architecture is required, optional, or not allowed for this form factor. Note that this refers to the operation of the device not the underlying silicon capabilities.

A form factor must provide the CLKREQ# signal if it supports L1 PM Substates. Form factor specifications must indicate if the CLKREQ# signal is required, optional, or not allowed.

Table 8-21 Form Factor Clocking Architecture Requirements §

| Clock Architecture | System Board (Motherboard) | Add-in Card (Module) | Retimer |
|--------------------|----------------------------|----------------------|---------|
| Common | ** | ** | ** |

| Clock Architecture | System Board (Motherboard) | Add-in Card (Module) | Retimer |
|--------------------|----------------------------|----------------------|---------|
| SRNS | ** | ** | ** |
| SRIS | ** | ** | ** |

** Each entry in the table must be filled in with one of: Required, Optional, or Not Allowed

If the Common Reference Clock architecture is required or optional for the form factor, then there must be an additional table (see § Table 8-22) providing details for the common clock. Each entry in the table is marked required, optional, not allowed, or NA. “Clock Source” indicates the source of the common reference clock, if applicable. “SSC” indicates whether the clock source is spread.

Table 8-22 Form Factor Common Clock Architecture Details §

| Common Clock Details | System Board (Motherboard) | Add-in Card (Module) | Retimer |
|----------------------|----------------------------|----------------------|---------|
| Clock Source | | | |
| SSC | | | |

If a form factor has clocking requirements that cannot be provided in this simple one or two table form then careful consideration must be given to ensure that the form factor requirements are supported by this specification.

As an example the populated tables are shown for a hypothetical form factor that requires all components use the common clock architecture and does not allow the use of any other clocking architecture (see § Table 8-23 and § Table 8-24). The common clock source is required to be provided by the motherboard component and may optionally have SSC. L1 PM Substates are not supported and therefore CLKREQ# is not defined as a connector signal for this example form factor.

Table 8-23 Form Factor Clocking Architecture Requirements Example §

| Clock Architecture | System Board (Motherboard) | Add-in Card (Module) | Retimer |
|--------------------|----------------------------|----------------------|-------------|
| Common | Required | Required | Required |
| SRNS | Not Allowed | Not Allowed | Not Allowed |
| SRIS | Not Allowed | Not Allowed | Not Allowed |

Table 8-24 Form Factor Common Clock Architecture Details Example §

| Common Clock Details | System Board (Motherboard) | Add-in Card (Module) | Retimer |
|----------------------|----------------------------|----------------------|-------------|
| Clock Source | Required | Not Allowed | Not Allowed |
| SSC | Optional | N/A | N/A |

It is important for form factor specifications to recognize that the CLKREQ# signal is required if L1 PM Substates are to be supported, and that for L1 PM Substates the CLKREQ# signal is used even if there is no common reference clock.

If a form factor has clocking requirements that cannot be provided in this simple one or two table form then careful consideration must be given to ensure that the form factor requirements are supported by this specification.

↑↓Single Root↓

9. I/O Virtualization and Sharing §

↑↑This chapter covers two approaches to virtualization and sharing, Single Root I/O Virtualization (SR-IOV) and Scalable I/O Virtualization (SIOV).↑ SR-IOV ↑↑standardizes more of the device virtualization scenario but requires more hardware per virtualized interface than SIOV. SIOV relies on system software to fill in more of the overall virtualization scenario, thus requiring less hardware per virtualized interface, with the intention that each virtualized interface will have a reduced cost and complexity.↑

ECN: Base 6.3
SIOV△↔

9.1 ↑↑IOV↑ Architectural Overview §

Within the industry, significant effort has been expended to increase the effective hardware resource utilization (i.e., application execution) through the use of virtualization technology. ↑↑IOV, including↑ Single Root I/O Virtualization and Sharing ↑↑(SR-IOV) enables↓ ↑↑(SR-IOV) and Scalable I/O Virtualization (SIOV) enable↑ multiple System Images (SI) to share PCI hardware resources.

ECN: Base
6.3 SIOV△↔

ECN: Base 6.3
SIOV△↔

To illustrate how this technology can be used to increase effective resource utilization, consider the generic platform configuration illustrated in § Figure 9-1 .

Base 6.4 vs Base 6.3

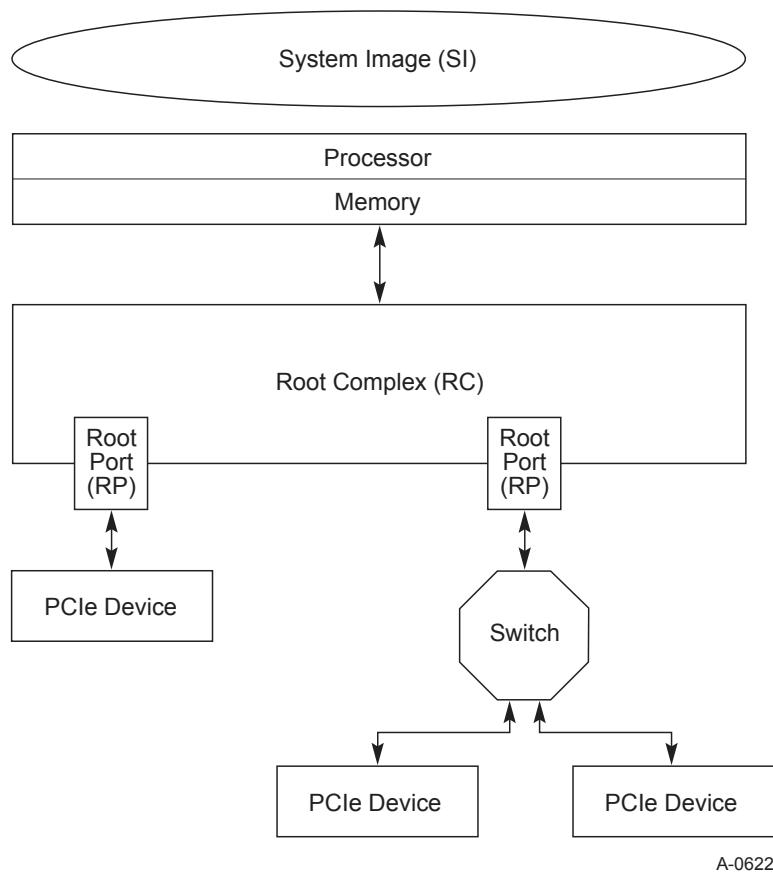


Figure 9-1 Generic Platform Configuration §

The generic platform configuration is composed of the following components:

- PCIe Root Complex (RC), which includes:
 - Processor - general purpose, embedded, or specialized processing element
 - Memory - general purpose or embedded
 - Root Complex Integrated Endpoints (RCiEPs)
 - PCIe Root Ports (RP) - Each RP represents a separate hierarchy.
- PCIe Switch - provides I/O fan-out and connectivity
 - PCIe Device - multiple I/O device types, (e.g., network, storage, etc.)
 - System Image - software such as an operating system that is used to execute applications or trusted services, (e.g., a shared or non-shared I/O device driver). ↑↑(VI)↑↑(VI)- often referred to as a hypervisor or virtual machine manager) ECN: Base 6.3
SIOV△↔

In order to increase the effective hardware resource utilization without requiring hardware modifications, multiple SI can be executed. Software termed a Virtualization Intermediary ↑↑(VI)↑↑(VI)- often referred to as a hypervisor or virtual machine manager) is interposed between the hardware and the SI as illustrated in § Figure 9-2 .

ECN: Base 6.3
SIOV△↔

Base 6.4 vs Base 6.3

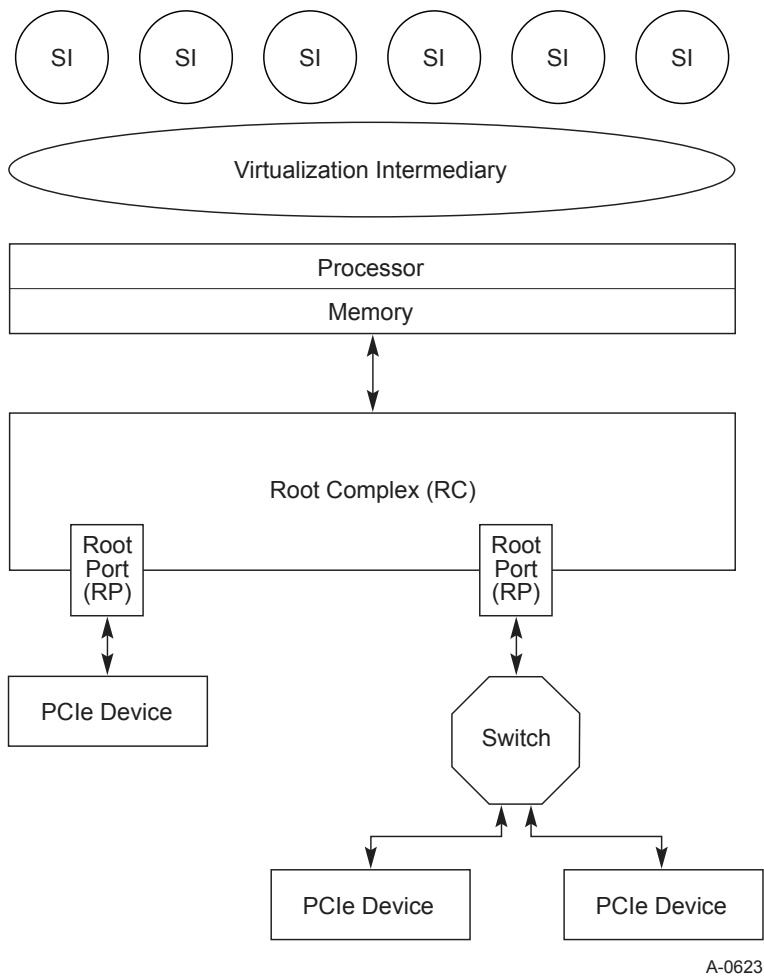


Figure 9-2 Generic Platform Configuration with a VI and Multiple SI §

The VI takes sole ownership of the underlying hardware. Using a variety of methods outside of the scope of this specification, the VI abstracts the hardware to present each SI with its own virtual system. The actual hardware resources available to each SI can vary based on workload or customer-specific policies. While this approach works well for many environments, I/O intensive workloads can suffer significant performance degradation. Each I/O operation - inbound or outbound - must be intercepted and processed by the VI adding significant platform resource overhead.

↑↓SR-IoV↓

↑↑IoV↑ provides tools to reduce these platform resources overheads. The benefits of ↑↓SR-IoV↓ ↑↑IoV↑ are:

ECN: Base 6.3
SIOV△◀▷

- The ability to eliminate VI involvement in main data movement actions - DMA, Memory space access, interrupt processing, etc. Elimination of VI interception and processing of each I/O operation can provide significant application and platform performance improvements.
- Standardized method to control ↑↓SR-IoV↓ ↑↑IoV↑ resource configuration and management through Single Root PCI Manager (SR-PCIM).

ECN: Base 6.3
SIOV△◀▷

Base 6.4 vs Base 6.3

- The ability to reduce the hardware requirements and associated cost with provisioning potentially a significant number of I/O Functions within a device.
- The ability to integrate SR-IOV with other I/O virtualization technologies such as Address Translation Services (ATS), **Address Translation and Protection Table (ATPT)** technologies, and interrupt remapping technologies to create robust, complete I/O virtualization solutions. **Together, the ATPT and interrupt remapping technologies are used by the Translation Agent, often referred to as an IOMMU, or I/O Memory Management Unit.**

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

§ Figure 9-3 illustrates an example SR-IOV capable platform.

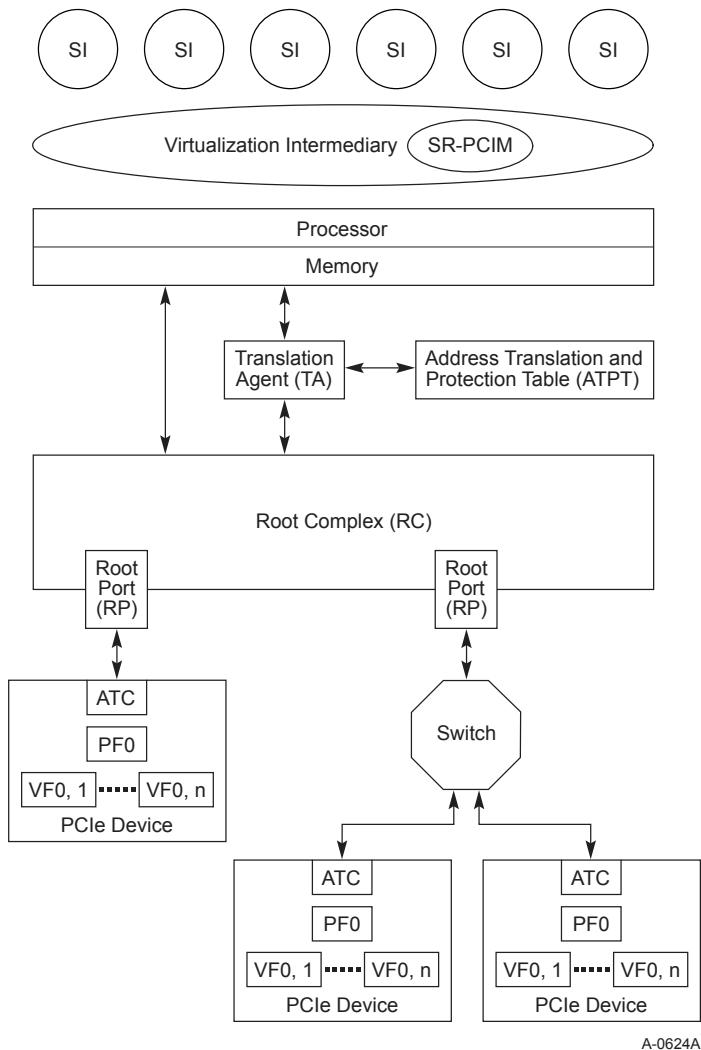


Figure 9-3 Generic Platform Configuration with SR-IOV and IOV Enablers §

The generic platform configuration is composed of the following additional functional elements:

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

- SR-PCIM - Software responsible for the configuration of the SR-IOV Extended Capability and SIOV Extended Capability,

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

Base 6.4 vs Base 6.3

- management of Physical Functions $\downarrow\downarrow$ and $\uparrow\uparrow$ Virtual Functions , and $\uparrow\uparrow$ SDIs , and \uparrow
- processing of associated error events and overall device controls such as power management and hot-plug services.
- Optional Translation Agent (TA) - A TA is hardware or a combination of hardware and software responsible for translating an address within a PCIe transaction into the associated platform physical address. A TA may contain an Address Translation Cache (ATC) to accelerate translation table access. A TA may also support Address Translation Services (ATS) which enables a PCIe Function to obtain address translations *a priori* to DMA access to the associated memory. See § Chapter 10. for more details on ATS benefits and operation.
- Optional Address Translation and Protection Table (ATPT) - An ATPT contains the set of address translations accessed by a TA to process PCIe requests - DMA Read, DMA Write, or interrupt requests. See Address Translation Services (§ Chapter 10.) for additional details.
 - In PCIe, interrupts are treated as memory write operations. Through the combination of a Requester Identifier and the address contained within a PCIe transaction, an interrupt can be routed to any target (e.g., a processor core) transparent to the associated I/O Function.
 - DMA Read and Write requests are translated through a combination of the Routing ID and the address contained within a PCIe transaction.
- Optional Address Translation Cache (ATC) - An ATC can exist in two locations within a platform - within the TA which can be integrated within or sit above an RC - or within a PCIe Device. Within an RC, the ATC enables accelerated translation look ups to occur. Within a Device, the ATC is populated through ATS technology. PCIe transactions that indicate they contain translated addresses may bypass the platform's ATC in order to improve performance without compromising the benefits associated with ATPT technology. See Address Translation Services (§ Chapter 10.) for additional details.
- Optional Access Control Services (ACS) - ACS defines a set of control points within a PCI Express topology to determine whether a TLP should be routed normally, blocked, or redirected. In a system that supports $\downarrow\downarrow$ SR-IOV, \downarrow $\uparrow\uparrow$ IOV, \uparrow ACS may be used to prevent device Functions $\uparrow\uparrow$ or SDIs \uparrow assigned to the VI or different SIs from communicating with one another or a peer device. Redirection may permit a Translation Agent to translate Upstream memory TLP addresses before a peer-to-peer forwarding decision is made. Selective blocking may be provided by the optional ACS P2P Egress Control. ACS is subject to interaction with ATS. See § Section 6.12 for additional details.

ECN: Base 6.3
 SIOV $\triangleleft\triangleright$
- Physical Function (PF) - A PF is a PCIe Function that supports the SR-IOV Extended Capability $\uparrow\uparrow$ SIOV Extended Capability \uparrow and is accessible to an SR-PCIM , a VI, or an SI.

ECN: Base 6.3
 SIOV $\triangleleft\triangleright$
- $\uparrow\uparrow$ Virtual Device Interface \rightarrow A $\downarrow\downarrow$ VF is a “light-weight” PCIe Function that $\uparrow\uparrow$ virtual device interface \uparrow is directly accessible by an SI.
 - $\downarrow\downarrow$ Minimally, resources \downarrow $\uparrow\uparrow$ Resources \uparrow associated with the $\downarrow\downarrow$ main \downarrow data $\downarrow\downarrow$ movement of \downarrow $\uparrow\uparrow$ processing in \uparrow the $\uparrow\uparrow$ Physical \uparrow Function are available to the SI. $\downarrow\downarrow$ Configuration resources should be \downarrow $\uparrow\uparrow$ Device configuration and control surfaces are typically \uparrow restricted to a trusted software component such as a VI or $\downarrow\downarrow$ SR PCIM \downarrow $\uparrow\uparrow$ SR-PCIM, which \downarrow interacts with the PF in a manner that is not exposed to an SI directly. \uparrow
 - A $\downarrow\downarrow$ VF \downarrow $\uparrow\uparrow$ virtual device interface \uparrow can be serially shared by different $\downarrow\downarrow$ SI, \downarrow $\uparrow\uparrow$ SIs, \uparrow (i.e., a $\downarrow\downarrow$ VF \downarrow $\uparrow\uparrow$ device Interface \uparrow can be assigned to one SI and then reset and assigned to another $\downarrow\downarrow$ SI). \downarrow $\uparrow\uparrow$ SI.) \uparrow
 - $\uparrow\uparrow$ Virtual Function (VF) \rightarrow A VF $\downarrow\downarrow$ can be optionally migrated from one PF to another PF. The migration process itself \downarrow is $\downarrow\downarrow$ outside \downarrow $\uparrow\uparrow$ a “light-weight” PCIe Function that is a virtual device interface. VFs are discovered and configured via \downarrow the $\downarrow\downarrow$ scope of this specification but \downarrow $\uparrow\uparrow$ SR-IOV Extended Capability structure. \uparrow

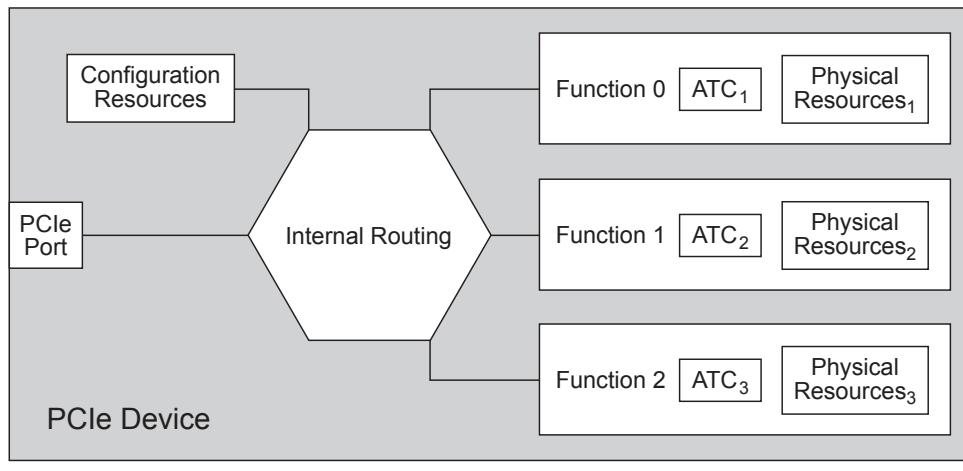
ECN: Base 6.3
 SIOV $\triangleleft\triangleright$

Base 6.4 vs Base 6.3

- **VFs have a Configuration Space that is facilitated through configuration controls defined within this specification.** **a subset of that of an unvirtualized Function with a Type 0 header.**
- **VFs expose registers and memory to an SI via a section of a Virtual Function BAR.**
- All VFs associated with a PF must be the same device type **and class** as the PF, (e.g., the same network device type or the same storage device type).
- **Scalable Device Interface (SDI) – An SDI is a virtual device interface exposed and configured via the SIOV Extended Capability structure in conjunction with SR-PCIM.**
 - **SDIs have no Configuration Space of their own, relying on SR-PCIM for the portions of interface management that would have been expressed as Capability Structures or in the Type 0 header if the Device Interface were implemented as a VF.**
 - **SDIs have no BAR of their own, relying on the VI to use page tables to map a virtual view of the SDI into an SI, where the pages mapped may be in the PF's BARs, in host RAM, or they may be entirely synthesized by the VI.**
 - **SDIs, as they do not express their device type and class through a Configuration Space, are not limited to exposing a device interface of the same type as the PF.**
 - **SDIs have a RID but do not meet the definition of a Function. Requirements in this specification for Functions do not apply to SDIs unless specifically called out.**

To compare and contrast a PCIe Device with a PCIe **SR-IOV** capable device, examine the following set of figures. § Figure 9-4 illustrates an example PCIe-compliant Device.

ECN: Base 6.3
SIOV△



A-0625

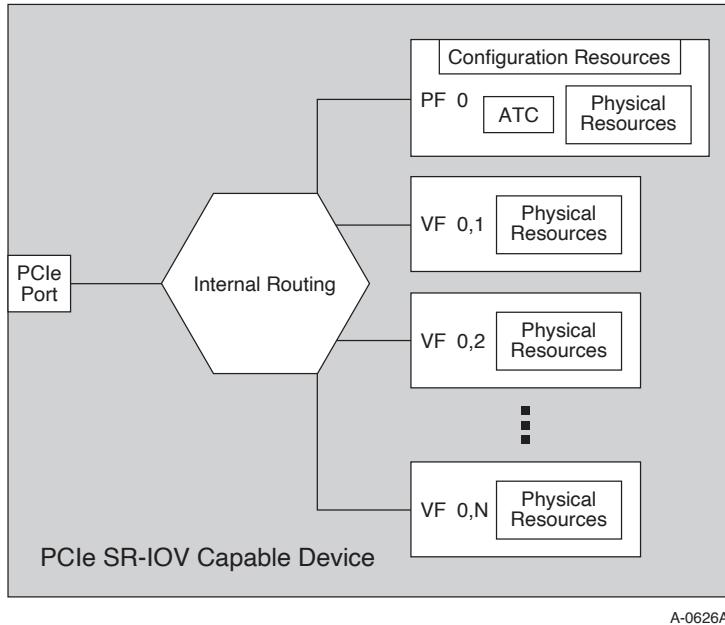
Figure 9-4 Example Multi-Function Device §

This figure illustrates an example multi-Function PCIe Device with the following characteristics:

- The PCIe Device shares a common PCIe Link. The Link and PCIe functionality shared by all Functions is managed through Function 0.
 - While this figure illustrates only three Functions, with the use of the Alternative Routing Identifier (ARI) capability, a PCIe Device can support up to 256 Functions.
 - All Functions use a single Bus Number captured through the PCI enumeration process.

- In this example, each PCIe Function supports the ATS capability and therefore has an associated ATC to manage ATS obtained translated addresses.
- Each PCIe Function has a set of unique physical resources including a separate configuration space and BAR.
- Each PCIe Function can be assigned to an SI. To prevent one SI from impacting another, all PCIe configuration operations should be intercepted and processed by the VI.

As this figure illustrates, the hardware resources scale with the number of Functions provisioned. Depending upon the complexity and size of the device, the incremental cost per Function will vary. To reduce the incremental hardware cost, a device can be constructed using SR-IOV to support a single PF and multiple VFs as illustrated in § Figure 9-5 .



A-0626A

Figure 9-5 Example SR-IOV Single PF Capable Device §

The example in § Figure 9-5 illustrates a single PF with N VFs. Key observations to note:

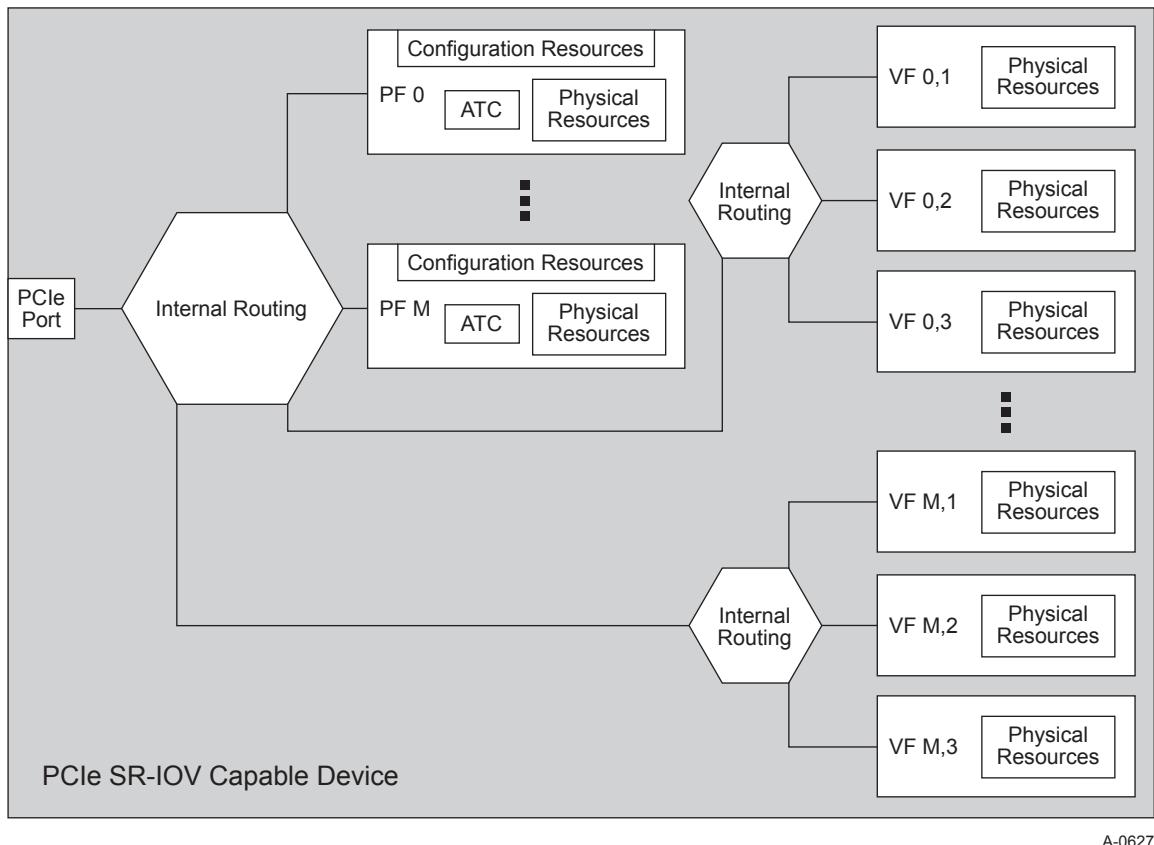
- The PF is a ~~PCIe-compliant~~ PCIe-compliant Function.
 - Initially and after a conventional reset, a PCIe Function that supports the SR-IOV capabilities defined in this specification shall have the SR-IOV capabilities disabled.
 - To discover the page sizes supported by a PF and its associated VF, the Supported Page Sizes configuration field is read. For additional information on how this field can be used to align PF or VF Memory space apertures on a system page boundary, see § Section 9.2.1.1 .
- PF nomenclature **PF M** designates the PF at Function number M.
- VF nomenclature **VF M,N** designates the Nth VF associated with PF M. VFs are numbered starting with 1 so the first VF associated with PF M is VF M,1.
- Each VF shares a number of common configuration space fields with the PF; (i.e., where the fields are applicable to all VF and controlled through a single PF. Sharing reduces the hardware resource requirements to implement each VF).
 - A VF uses the same configuration mechanisms and header types as a PF.

- All VFs associated with a given PF share a VF BAR set (see § [Section 9.4.3.14](#)) and share a VF Memory Space Enable (MSE) bit in the SR-IOV extended capability (see § [Section 9.4.3.3.4](#)) that controls access to the VF Memory space. That is, if the VF MSE bit is Clear, the memory mapped space allocated for all VFs is disabled.
- The InitialVFs and TotalVFs fields (see § [Section 9.4.3.5](#) and § [Section 9.4.3.6](#)) are used to discover the maximum number of VFs that can be associated with a PF.
- TotalVFs and InitialVFs shall contain the same value²⁰⁵.
- Each Function, PF, and VF is assigned a unique Routing ID. The Routing ID for each PF is constructed as per § [Section 2.2.4.2](#). The Routing ID for each VF is determined using the Routing ID of its associated PF and fields in that PF's [SR-IOV Extended Capability](#).
- All PCIe and SR-IOV configuration access is assumed to be through a trusted software component such as a VI or an [SR-PCIM](#).
- Each VF contains a non-shared set of physical resources required to deliver Function-specific services, (e.g., resources such as work queues, data buffers, etc.) These resources can be directly accessed by an SI without requiring VI or [SR-PCIM](#) intervention.
- One or more VF may be assigned to each SI. Assignment policies are outside the scope of this specification.
- While this example illustrates a single ATC within the PF, the presence of any ATC is optional. In addition, this specification does not preclude an implementation from supporting an ATC per VF within the Device.
- Internal routing is implementation specific.
- While many potential usage models exist regarding PF operation, a common usage model is to use the PF to bootstrap the device or platform strictly under the control of a VI. Once the [SR-IOV Extended Capability](#) is configured enabling VF to be assigned to individual SI, the PF takes on a more supervisory role. For example, the PF can be used to manage device-specific functionality such as internal resource allocation to each VF, VF arbitration to shared resources such as the PCIe Link or the Function-specific Link (e.g., a network or storage Link), etc. These policy, management, and resource allocation operations are outside the scope of this specification.

Another example usage model is illustrated in § [Figure 9-6](#). In this example, the device supports multiple PFs each with its own set of VFs.

²⁰⁵ InitialVFs and TotalVFs could have had different values due to MR-IOV (now deprecated).

Base 6.4 vs Base 6.3



A-0627

Figure 9-6 Example SR-IOV Multi-PF Capable Device §

Key observations to note:

- Each PF can be assigned ~~1↓zero↓~~ ~~1↑one↑~~ or more VFs. The number of VFs per PF is not required to be identical for all PFs within the device.
- The ARI Extended Capability enables Functions to be assigned to Function Groups and defines how Function Group arbitration can be configured. PFs and VFs can be assigned to Function Groups and take advantage of the associated arbitration capabilities. Within each Function Group, though, arbitration remains implementation specific.
- Internal routing between PFs and VFs is implementation specific.
- For some usage models, all PFs may be the same device type; (e.g., all PFs deliver the same network device or all deliver the same storage device functionality). For other usage models, each PF may represent a different device type; (e.g., in § Figure 9-6 , one PF might represent a network device while another represents an encryption device).
 - In situations where there is a usage model dependency between device types, such as for each VF that is a network device type, each SI also requires a VF that is an encryption device type. The SR-IOV Extended Capability provides a method to indicate these dependencies. The policies used to construct these dependencies as well as assign dependent sets of VF to a given SI are outside the scope of this specification.

ECN: Base 6.3
SIOV△↔

As seen in the prior example, the number of PF and VF can vary based on usage model requirements. To support a wide range of options, an **↓↑SR-IOV Device↓↑↑SR-IOV Device↑** can support the following number and mix of PF and VF:

- Using the Alternative Routing Identifier (ARI) capability, a device may support up to 256 PFs. Function Number assignment is implementation specific and may be sparse throughout the 256 Function Number space.
- A PF can only be associated with the Device's captured Bus Number as illustrated in § [Figure 9-7](#).
- SR-IOV Devices may consume more than one Bus Number. A VF can be associated with any Bus Number within the Device's Bus Number range - the captured Bus Number plus any additional Bus Numbers configured by software. See § [Section 9.2.1.2](#) for details.
 - Use of multiple Bus Numbers enables a device to support a very large number of VFs - up to the size of the Routing ID space minus the bits used to identify intervening **↓↑busses↓↑↑buses↑**
 - If software does not configure sufficient additional Bus Numbers, then the VFs implemented for the additional Bus Numbers may not be visible.

IMPLEMENTATION NOTE: FUNCTION CO-LOCATION §

The ARI Extended Capability enables a Device to support up to 256 Functions - Functions, PFs, or VFs in any combination - associated with the captured Bus Number. If a usage model does not require more than 256 Functions, implementations are strongly encouraged to co-locate all Functions, PFs, and VFs within the captured Bus Number and not require additional Bus Numbers to access VFs.

Base 6.4 vs Base 6.3

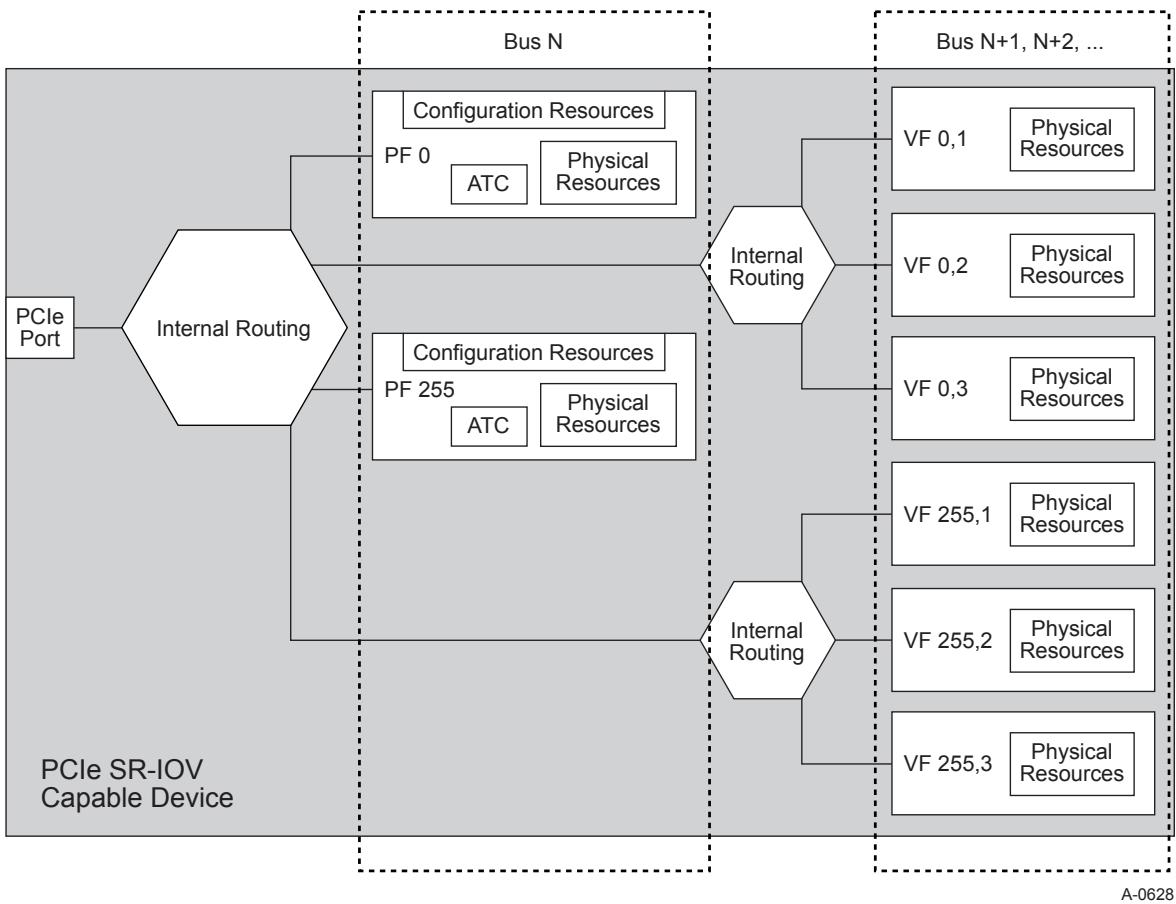


Figure 9-7 Example ~~↓↓↓SR-IOV Device↓~~ ~~↑↑↑SR-IOV Device↑~~ with Multiple Bus Numbers §

In this ~~↓↓↓last↓~~ example, § Figure 9-8 , a device implementation may mix any number of Functions, PFs, and VFs.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

Base 6.4 vs Base 6.3

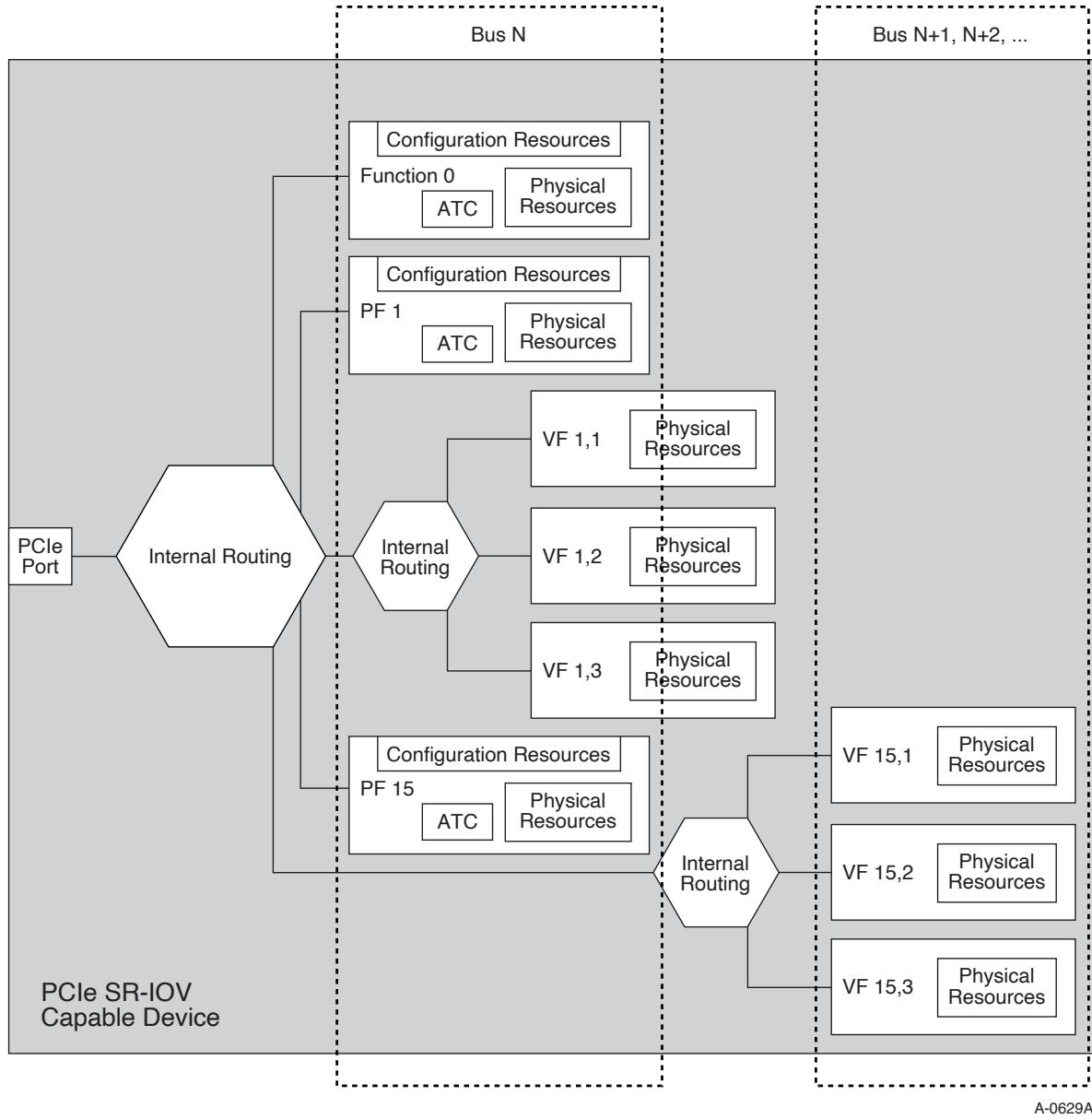


Figure 9-8 Example ~~↑↓SR-IOV Device~~ ~~↑↑SR-IOV Device~~ with a Mixture of Function Types §

Key observations to note:

- Each Device must contain a Function 0. Function 0 may be a PF (i.e., it may include the SR-IOV extended capability).
- Any mix of Functions can be associated with the captured Bus Number.
 - Non-VFs can only be associated with the captured Bus Number.
- If the ARI Extended Capability is supported, Functions can be assigned to Function Groups. The assignment policy is outside the scope of this specification. If the ARI Extended Capability is not supported, Functions can still use the Function arbitration capabilities as defined in § Section 6.3.3.4.

Base 6.4 vs Base 6.3

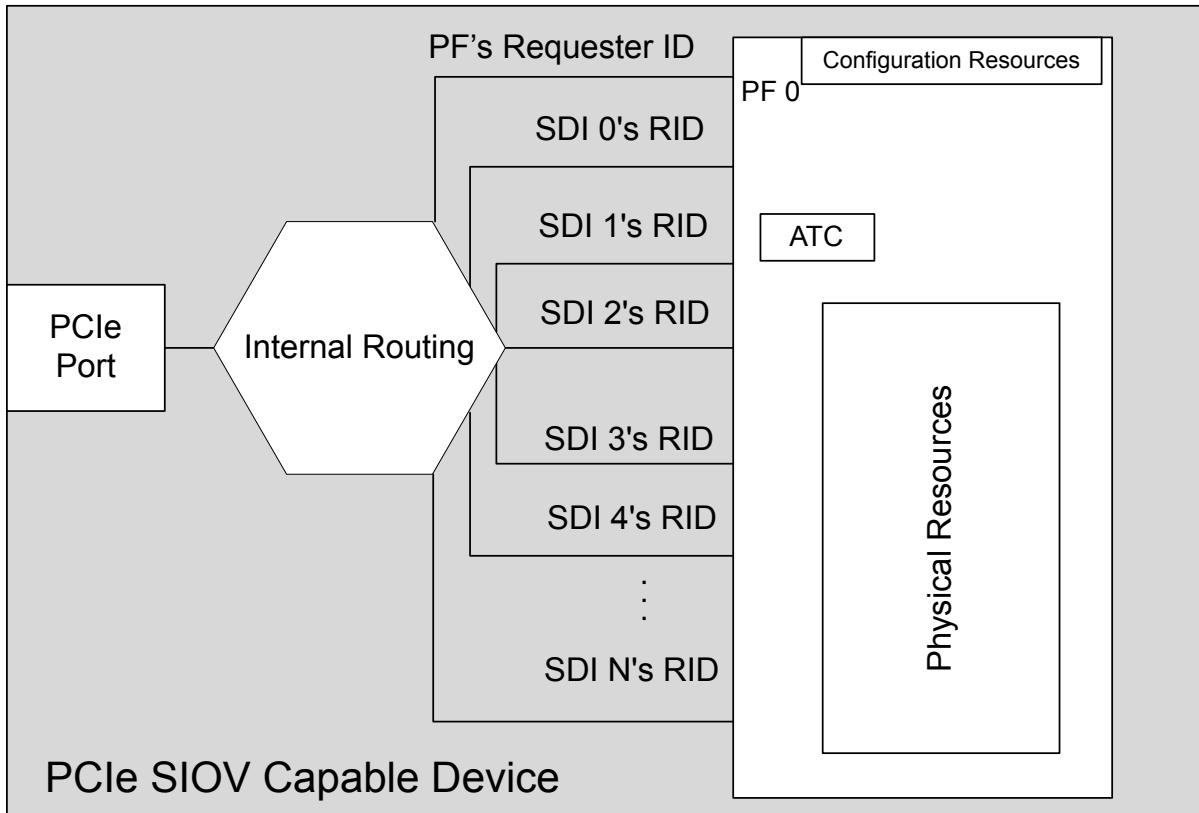


Figure 9-9 Example SIOV Capable Device

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

The example in § Figure 9-9 shows a device implementing Scalable IOV, contrasting § Figure 9-5. It illustrates a similar device but using Scalable Device Interfaces instead of Virtual Functions. For the purposes of the example, assume that the SDIs are in use by SIs running on top of a VI.↑

Key observations to note:↑

- The PF is a Function.↑
 - Initially and after a conventional reset, a PCIe Function that supports Scalable IOV shall have the SIOV capabilities disabled.↑
 - Discovery of page sizes supported by the PF is analogous to SR-IOV.↑
- All SDI Configuration Space access is assumed to be fully virtualized by a VI.↑
- Each SDI is a composition of physical resources within the PF.↑
 - Virtual BARs, as viewed from an SI, are mappings onto sets of pages within the PF BAR, if the SI requires the SDI to be presented by the VI as a Function. Ranges within the virtual BAR may also be provided entirely by the VI, using SR-PCIM to handle reads and writes.↑
 - Virtual MSI or MSI-X is a mapping onto the internal interrupt management logic, if the SDI generates interrupts. Presentation of an MSI Capability or an MSI-X Capability to an SI may leverage SR-PCIM, instead of being implemented in device hardware or firmware.↑
 - Device-specific resources can be either dedicated to an SDI or shared across SDIs.↑
- Traffic on behalf of a specific SDI is tagged with a RID dedicated to that SDI.↑

IMPLEMENTATION NOTE: SDI MSI STATE MANAGEMENT §

↑↑Scalable Device Interfaces may need to generate interrupts, and when they do, the SDIs have the same underlying requirements that any Function would. The device should be able to store the state implied by the interrupt model underlying MSI or the interrupt model underlying MSI-X. The device should store this state somewhere, but this specification does not stipulate where. SR-PCIM handles mapping the SI's interrupt requirements onto device-internal state.↑

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

↑↑One potential strategy for this is to map MSI-X interrupt configurations from the SI onto MSI-X table entries in the PF's BAR, where an Interrupt Message will be sent using the contents of a specific entry, using the SDI's RID. Other strategies might involve storing state equivalent to MSI or MSI-X in different parts of the device.↑

IMPLEMENTATION NOTE: SHARING PAGES IN THE PF BAR WITH MULTIPLE SDIS §

↑↑SR-PCIM may choose to share pages of PF BAR space with multiple SDIs, but this comes with restrictions. If a page is used with multiple SDIs, the device will have no indication of which SDI is being read or written, since reads and writes may be address-routed and may not carry the RID associated with the SDI. Any side effects of a read or a write that would cause traffic to be initiated by the device will not be able to determine the RID of the associated SDI .↑

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

↑↑Devices implementing TDISP face additional complexity, as some accesses may originate from within a Trusted Execution Environment (TEE), while others originate from outside it. Moreover, TSMs and VIs may not support mapping common pages into TVMs .↑

9.1.1 ↑↑RID in Endpoints where a PF contains an SIOV Extended Capability Structure §

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

↑↑To allow SDIs to be used within the context of separate address spaces, an Endpoint employing SIOV uses a unique RID for each SDI. This RID must contain a Bus Number that has been allocated to the Endpoint device containing the PF with an SIOV Extended Capability structure. An SDI's RID must not be used by any Function or any other SDI .↑

↑↑SR-IOV assigns a Virtual Function a RID based on a formula where an offset and a stride is used to compute the RID of the VF. This is an option with SIOV, though it may not allow the device to efficiently use all of the RID space, perhaps because the upstream bridges are Flattening Portal Bridges. Similarly, some PCIe Switches can be attached to multiple Root Complexes, simultaneously (though this is a vendor-specific extension, and when this technique is employed, traffic is often routed to one Root Complex or another based on partitioning the RID space, perhaps by Bus Number). In these cases, maximizing the number of available SDIs involves using all available RIDs, which requires a dynamic association between RIDs and SDIs.↑

↑↑This choice is expressed through the SIOV Extended Capability structure. The first option allows the device to share or reuse RID mapping logic from existing SR-IOV devices. Just as with SR-IOV, the SDIs RID is calculated by multiplying the SDI number by a supplied stride and then applying an offset to the PF's RID.↑

↑↑↑The other option is to allow the VI to supply the RID for the SDI. If this is employed, after Bus Numbers have been assigned to all the relevant bridges during the boot process and before any SDIs are placed into use, the set of potential RIDs can be calculated with the following algorithm:↑

1. ↑↑↑Identify the set of Bus Numbers passed through the upstream bridge by starting with its Secondary and Subordinate Bus Numbers and adding in all buses implied by the configuration of a Flattening Portal Bridge capability, if present.↑
2. ↑↑↑Calculate the set of RIDs available to the Endpoint on each of these logical buses. Each bus has 256 RIDs available.↑
3. ↑↑↑Remove from this set of RIDs any Function enumerable by reading Configuration Space of Function 0 on the Secondary Bus and continuing in order as implied by the Multi-Function Device bit in its Header Type register.↑
4. ↑↑↑Further remove from this set of RIDs any Function enumerable by the Alternative Routing Interpretation (ARI).↑
5. ↑↑↑Find each of the PFs within this Endpoint, other than the one for which this algorithm is targeting.↑
 - a. ↑↑↑If that PF has an SR-IOV Extended Capability structure and no Scalable IOV Extended Capability structure, calculate the set of RIDs used by its VFs and remove them from the target set.↑
 - b. ↑↑↑If that PF has a Scalable IOV Extended Capability structure and it uses the Strided RID mapping, calculate the set of RIDs it would use for SDIs and remove them from the target set.↑
 - c. ↑↑↑If that PF has an SIOV Extended Capability and it does not use the Strided RID mapping, remove any RIDs currently in use by it.↑
6. ↑↑↑Find all Phantom Functions and Shadow Functions and remove them from the set of RIDs.↑

↑↑↑The resulting set of RIDs contains valid values for use by SDIs.↑

9.1.2 ↑↑↑SDI Reset §

ECN: Base 6.3
SIOV△◀▷

↑↑↑The SDI reset mechanism enables software to quiesce and reset the SDI functionality. Three example usage models illustrate the need for this feature:↑

- ↑↑↑In some systems, it is possible that the software entity that controls the SDI will cease to operate normally. To prevent data corruption, it is necessary to stop all PCI Express and external I/O (not PCI Express) operations being performed by the SDI.↑
- ↑↑↑In a partitioned environment where the SDI is migrated from one partition to another, it is necessary to ensure that no residual “knowledge” of the prior partition be retained by the SDI, for example, a user’s secret information entrusted to the first partition but not to the second.↑
- ↑↑↑When system software is taking down the software stack for the SDI and then rebuilding that stack, it is sometimes necessary to return the state to an uninitialized state before rebuilding the SDI’s software stack.↑

↑↑↑Each SDI must be independently resettable without affecting the operation of other SDIs. Reset of an SDI is performed through device-specific interfaces. The SDI reset causes the device to abort (discard) all in flight and accepted operations to the SDI by specific domain and reset SDI specific configuration on the device to a known state:↑

- ↑↑↑Unarchitected SDI state configured through the PF or SR-PCIM interface is permitted to persist across SDI instantiations.↑
- ↑↑↑Architected SDI state configured through the PF or SR-PCIM interface is permitted to be restored to earlier values or set to new values. That is, when the SDI is enabled again, the new instance of SDI is permitted to have a different set of architected capabilities and/or their attributes (e.g., RID, Memory Space resources, interrupt message resources) from the previous instance.↑

↑↑↑The SDI reset is conceptually equivalent to an FLR. In response to the SDI reset,↑

- ↑↑↑The SDI must not give the appearance of an initialized adapter with an active host on any external interfaces controlled by that SDI. The steps needed to terminate activity on external interfaces are outside of the scope of this specification.↑
 - ↑↑↑For example, a network adapter must not respond to queries that would require adapter initialization by the host system or interaction with an active host system, but is permitted to perform actions that it is designed to perform without requiring host initialization or interaction.↑
- ↑↑↑The SDI must not retain within itself software readable state that potentially includes secret information associated with any preceding use of the SDI. Main host memory assigned to the SDI must not be modified by the SDI.↑
 - ↑↑↑For example, an SDI with internal memory readable directly or indirectly by host software must clear or randomize that memory.↑

↑↑↑An SDI reset must ensure that the reset is not reported as complete until all of the following conditions are satisfied:↑

- ↑↑↑All DMA write operations by the SDI are drained or aborted.↑
- ↑↑↑All DMA read operations by the SDI have completed or aborted.↑
- ↑↑↑All interrupts from the SDI have been generated.↑
- ↑↑↑If the SDI is capable of Address Translation Service (ATS), all ATS requests by the SDI have completed or aborted.↑
- ↑↑↑If the SDI is capable of Page Request Service (PRS), no more page requests will be generated by the SDI. Additionally, either page responses have been received for all page requests generated by the SDI or the SDI will discard page responses for any outstanding page requests by the SDI.↑

↑↑↑While the SDI reset is in progress:↑

- ↑↑↑If a Request arrives, the Request is permitted to be silently discarded (following update of flow control credits) without logging or signaling it as an error.↑
- ↑↑↑If a Completion arrives, the Completion is permitted to be handled as an Unexpected Completion or to be silently discarded (following update of flow control credits) without logging or signaling it as an error.↑

9.2 SR-IOV Initialization and Resource Allocation §

The following sections describe how software determines that a Device is SR-IOV capable and subsequently identifies VF resources through Virtual Function Configuration Space.

9.2.1 SR-IOV Resource Discovery §

This section describes the fields that must be configured before enabling a PF's IOV Capabilities. The VFs are enabled by Setting the PF's VF Enable bit (see § Section 9.4.3.3.1) in the SR-IOV extended capability.

The NumVFs field (see § Section 9.4.3.7) defines the number of VFs that are enabled when VF Enable is Set in the associated PF.

9.2.1.1.1 Configuring the VF BAR Mechanisms §

This section describes how the VF BARs are configured to map memory space. VFs do not support I/O Space and thus VF BARs shall not indicate I/O Space.

The System Page Size field (see § Section 9.4.3.13) defines the page size the system will use to map the VF's PCIe memory addresses when the PF's IOV Capabilities are enabled. The System Page Size field is used by the PF to align the Memory space aperture defined by each VF BAR to a system page boundary. The value chosen for the System Page Size must be one of the Supported Page Sizes (see § Section 9.4.3.12) in the SR-IOV extended capability.

The behavior of VF BARs is the same as the normal PCI Memory Space BARs (see § Section 7.5.1.2.1), except that a VF BAR describes the aperture for each VF, whereas a PCI BAR describes the aperture for a single Function. The attributes for some of the bits in the VF BARs are affected by the VF Resizable BAR Extended Capability (see § Section 7.8.7) if it is implemented.

- The behavior described in § Section 7.5.1.2.1 for determining the memory aperture of a Function's BAR applies to each VF BAR. That is, the size of the memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR. The results read back must be interpreted as described in § Section 7.5.1.2.1.
- The behavior for assigning the starting memory space address of each BAR associated with the first VF is also as described in § Section 7.5.1.2.1. That is, the address written into each VF BAR is used by the Device for the starting address of the first VF.
- The difference between VF BARs and BARs described in § Section 7.5.1.2.1 is that for each VF BAR, the memory space associated with the second and higher VFs is derived from the starting address of the first VF and the memory space aperture. For any given VF_v, the starting address of its Memory space aperture for any implemented BAR_b) is calculated according to the following formula:

$$\text{BAR}_b \text{ VF}_v \text{ starting address} = \text{VF BAR}_b + (v - 1) \times (\text{VF BAR}_b \text{ aperture size})$$

where VF BAR_b aperture size is the size of VF BAR_b as determined by the usual BAR probing algorithm as described in § Section 9.4.3.14.

VF memory space is not enabled until both VF Enable and VF MSE have been Set (see § Section 9.4.3.3.1 and § Section 9.4.3.3.4). Note that changing System Page Size (see § Section 9.4.3.13) may affect the VF BAR aperture size.

§ Figure 9-10 shows an example of the PF and VF Memory space apertures.

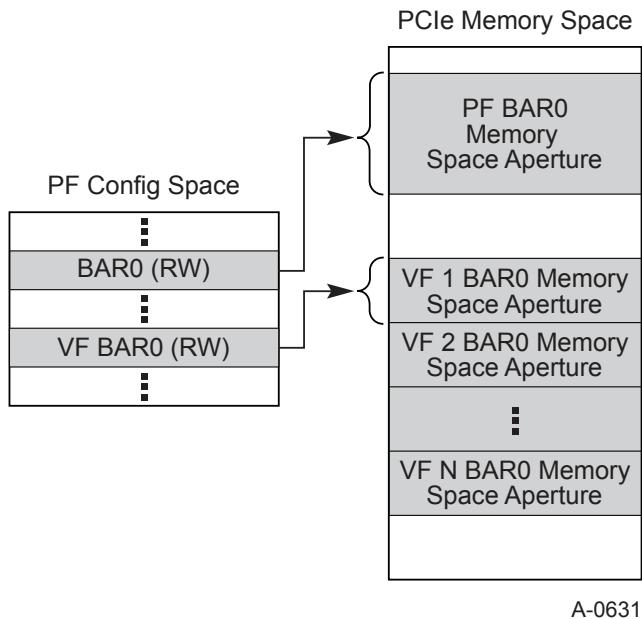


Figure 9-10 BAR Space Example for Single BAR Device §

9.2.1.2 VF Discovery §

The First VF Offset and VF Stride fields in the SR-IOV extended capability are 16-bit Routing ID offsets. These offsets are used to compute the Routing IDs for the VFs with the following restrictions:

- The value in NumVFs in a PF (§ Section 9.4.3.7) may affect the values in First VF Offset (§ Section 9.4.3.9) and VF Stride (§ Section 9.4.3.10) of that PF.
- The value in ARI Capable Hierarchy (§ Section 9.4.3.3.5) in the lowest-numbered PF of the Device (for example PF₀) may affect the values in First VF Offset and VF Stride in all PFs of the Device.
- NumVFs of a PF may only be changed when VF Enable (§ Section 9.4.3.3.1) of that PF is Clear.
- ARI Capable Hierarchy (§ Section 9.4.3.3.5) may only be changed when VF Enable is Clear in all PFs of a Device.

IMPLEMENTATION NOTE: NUMVFS AND ARI CAPABLE HIERARCHY §

After configuring NumVFs and ARI Capable Hierarchy where applicable, software may read First VF Offset and VF Stride to determine how many Bus Numbers would be consumed by the PF's VFs. The additional Bus Numbers, if any, are not actually used until VF Enable is Set.

§ Table 9-1 describes the algorithm used to determine the Routing ID associated with each VF.

Table 9-1 VF Routing ID Algorithm §

| VF Number | VF Routing ID |
|----------------------|--|
| VF 1 | (PF Routing ID + <u>First VF Offset</u>) Modulo 2^{16} |
| VF 2 | (PF Routing ID + <u>First VF Offset</u> + <u>VF Stride</u>) Modulo 2^{16} |
| VF 3 | (PF Routing ID + <u>First VF Offset</u> + $2 * \text{VF Stride}$) Modulo 2^{16} |
| ... | ... |
| VF N | (PF Routing ID + <u>First VF Offset</u> + $(N-1) * \text{VF Stride}$) Modulo 2^{16} |
| ... | ... |
| VF NumVFs (last one) | (PF Routing ID + <u>First VF Offset</u> + $(\text{NumVFs} - 1) * \text{VF Stride}$) Modulo 2^{16} |

All arithmetic used in this Routing ID computation is 16-bit unsigned dropping all carries.

All VFs and PFs must have distinct Routing IDs. The Routing ID of any PF or VF must not overlap with the Routing ID of any other PF or VF given any valid setting of NumVFs across all PFs of a device.

VF Stride and First VF Offset are constants. Except as stated earlier in this section, their values may not be affected by settings in this or other Functions of the Device.

VFs may reside on different Bus Number(s) than the associated PF. This can occur if, for example, First VF Offset has the value 0100h. A VF shall not be located on a Bus Number that is numerically smaller than its associated PF. A VF that is located on the same Bus Number as its associated PF shall not be located on a Device Number that is numerically smaller than the PF²⁰⁶.

VFs of an SR-IOV RCiEP Device are associated with the same Root Complex Event Collector (if any) as their PF. Such VFs are not reported in the Root Complex Event Collector Endpoint Association Extended Capability of the Root Complex Event Collector.

As per § Section 2.2.6.2, SR-IOV capable Devices that are associated with an Upstream Port capture the Bus Number from any Type 0 Configuration Write Request. SR-IOV capable Devices do not capture the Bus Number from any Type 1 Configuration Write Requests. SR-IOV capable RCiEPs use an implementation specific mechanism to assign their Bus Numbers.

Note: Bus Numbers are a constrained resource. Devices are strongly encouraged to avoid leaving “holes” in their Bus Number usage to avoid wasting Bus Numbers.

All PFs must be located on the Device’s captured Bus Number.

206. SR-IOV Devices immediately below a Downstream Port always have a Device Number of 0 and thus always satisfy this condition.

IMPLEMENTATION NOTE: VFS SPANNING MULTIPLE BUS NUMBERS §

As an example, consider an $\downarrow\downarrow$ SR-IOV Device $\uparrow\uparrow$ SR-IOV Device that supports a single PF. Initially, only PF 0 is visible. Software Sets ARI Capable Hierarchy . From the SR-IOV Extended Capability it determines: InitialVFs is 600, First VF Offset is 1 and VF Stride is 1.

- If software sets NumVFs in the range [0 … 255], then the Device uses a single Bus Number.
- If software sets NumVFs in the range [256 … 511], then the Device uses two Bus Numbers.
- If software sets NumVFs in the range [512 … 600], then the Device uses three Bus Numbers.

PF 0 and VF 0,1 through VF 0,255 are always on the first (captured) Bus Number. VF 0,256 through VF 0,511 are always on the second Bus Number (captured Bus Number plus 1). VF 0,512 through VF 0,600 are always on the third Bus Number (captured Bus Number plus 2).

Software should configure Switch Secondary and Subordinate Bus Number fields to route enough Bus Numbers to the Device. If sufficient Bus Numbers are not available, software should reduce a Device's Bus Number requirements by not enabling SR-IOV and/or reducing NumVFs for some or all PFs of the Device prior to enabling SR-IOV.

After VF Enable is Set in some PF n , the Device must Enable VF $n,1$ through VF n,m (inclusive) where m is the smaller of InitialVFs and NumVFs . A Device receiving a Type 0 Configuration Request targeting an Enabled VF located on the captured Bus Number must process the Request normally. A Device receiving a Type 1 Configuration Request targeting an Enabled VF not located on the captured Bus Number must process the Request normally. A Device receiving a Type 1 Configuration Request targeting the Device's captured Bus Number must follow the rules for handling Unsupported Requests. Additionally, if VF MSE is Set, each Enabled VF must respond to PCIe Memory transactions addressing the memory space associated with that VF.

Functions that are not enabled (i.e., Functions for VFs above m) do not exist in the PCI Express fabric. As per § Section 2.3.1 , addressing Functions that do not exist will result in Unsupported Request (UR) being returned. This includes Functions on additional Bus Numbers.

IMPLEMENTATION NOTE: MULTI-FUNCTION DEVICES WITH PFS AND SWITCH FUNCTIONS §

$\downarrow\downarrow$ SR-IOV devices $\uparrow\uparrow$ SR-IOV Device may consume multiple bus numbers. Additional bus numbers beyond the first one are consecutive and immediately follow the first bus number assigned to the device. If an $\downarrow\downarrow$ SR IOV device $\uparrow\uparrow$ SR-IOV Device also contains PCI-PCI Bridges (with Type 1 Configuration Space Headers), the SR-IOV usage must be accounted for when programming the Secondary Bus Number for those Bridges. Software should determine the last Bus Number used by VFs first and then configure any co-located Bridges to use Bus Numbers above that value.

9.2.1.3 Function Dependency Lists §

PCI Devices may have vendor-specific dependencies between Functions. For example, Functions 0 and 1 might provide different mechanisms for controlling the same underlying hardware. In such situations, the Device programming model might require that these dependent Functions be assigned to SIs as a set.

Function Dependency Lists are used to describe these dependencies (or to indicate that there are no Function dependencies). ↑↓Software ↑↑When using SR-IOV, software should assign PFs and VFs to SIs such that the dependencies are satisfied. ↑↑Function Dependency Lists cannot be used with PFs when using SIOV.

 ECN: Base 6.3
 SIOV $\triangleleft\triangleright$

See § Section 9.4.3.8 for details.

9.2.1.4 Interrupt Resource Allocation §

PFs and VFs support either MSI, MSI-X interrupts, or both if interrupt resources are allocated. VFs shall not implement INTx. MSI and MSI-X interrupts are described in § Section 6.1.4 .

For MSI-X interrupts, special address range isolation requirements apply for the MSI-X structures in PF and VF MMIO regions. See in § Section 7.7.2 .

9.2.2 SR-IOV Reset Mechanisms §

This section describes how reset mechanisms defined in § Section 6.6 affect Devices that support SR-IOV. It also describes the mechanisms used to reset a single VF and a single PF with its associated VFs.

9.2.2.1 SR-IOV Conventional Reset §

A Conventional Reset to a Device that supports SR-IOV shall cause all Functions (including both PFs and VFs) to be reset to their original, power-on state as per the rules in § Section 6.6.1 . § Section 9.4 describes the behavior for the fields defined.

Note: Conventional Reset clears VF Enable in the PF. Thus, VFs no longer exist after a Conventional Reset.

9.2.2.2 FLR That Targets a VF §

VFs must support Function Level Reset (FLR).

Note: Software may use FLR to reset a VF. FLR to a VF affects a VF's state but does not affect its existence in PCI Configuration Space or PCI Bus address space. The VFs BARn values (see § Section 9.4.3.14) and VF MSE (see § Section 9.4.3.3.4) in the PF's SR-IOV extended capability, and the VF Resizable BAR capability values (see § Section 7.8.7) are unaffected by FLRs issued to the VF.

9.2.2.3 FLR That Targets a PF §

PFs must support FLR .

FLR to a PF resets the PF state as well as the SR-IOV extended capability including VF Enable which means that VFs no longer exist.

9.2.3 IOV Re-initialization and Reallocation §

If VF Enable is Cleared after having been Set, all of the VFs associated with the PF no longer exist and must no longer issue PCIe transactions or respond to Configuration Space or Memory Space accesses. VFs must not retain any architected state after VF Enable has been Cleared (including sticky bits). For security, unarchitected VF state configured through the VF must be cleared or randomized, with the exception of persistent storage data.

Unarchitected VF state configured through the PF or SR-PCIM interface is permitted to persist across VF instances.

Architected VF state configured through the PF or SR-PCIM interface is permitted to be restored to earlier values or set to new values. That is, when VF Enabled is Set again, instances of VFs are permitted to have a different set of architected capabilities and/or their attributes (e.g., MSI-X Table size) from previous instances.

9.3 ↑↑SIOV Initialization and Resource Allocation↑ §

ECN: Base
6.3 SIOV△↔

9.3.1 ↑↑SIOV Resource Discovery↑ §

↑↑The following sections describe how software determines that a Device is SIOV capable and outlines the process for enabling, configuring and managing SDIs.↑

9.3.1.1 ↑↑Configuring SIOV Capabilities↑ §

↑↑In contrast to SR-IOV, SIOV hardware does not expose Configuration Space for each SDI. As such, there is no analogous configuration of the device's capabilities. Each individual SDI may be separately configured, enabled and disabled by the VI/SR-PCIM via an implementation-specific mechanism. If system software wishes to disable the PF and all the SDIs, it can use a conventional reset or a Function-Level Reset for this.↑

↑↑The number of available SDIs, and thus the number of Bus Numbers required for the Endpoint device, can be derived from the SIOV Extended Capability structure.↑

9.3.1.1.1 ↑↑Configuring an SDI↑ §

↑↑SIOV minimizes the complexity of a device by allowing device designers to implement internal partitioning and virtualization in ways that rely on internal knowledge of the device's design. SR-PCIM is expected to include device-specific extensions to accommodate discovery and configuration. Since SRPCIM itself isn't architected within this specification, and since the internal structure of various VIs differs quite notably, these extensions are not defined in this document.↑

9.3.1.1.1.1 ↑↑Behavior of a VI↑ §

ECN: Base 6.3
SIOV△↔

↑↑To support SIOV, a VI provides several sorts of mechanisms. The exact composition of these differs between VI implementations. They are listed here not to specify them, but to allow device designers to understand which mechanisms can be depended upon. Without any notion of what a VI might provide, however, device designers would be forced to implement many things in hardware, perhaps increasing the cost and complexity.↑

- ↑↑↑A VI provides a mechanism for a device-specific extension to SR-PCIM.↑

↑↑↑Logically, this mechanism is an extension of the device driver for the device, though it may be packaged separately from the device driver and it may run in a different process context or even in a different SI.↑

- ↑↑↑The device-specific extension to SR-PCIM provides read and write mechanisms for Configuration Space.↑

↑↑↑When the SDI is to be presented as a Function to an SI, these mechanisms, along with other parts of SR-PCIM, allow the SI to interact with the SDI as if it were a Function with a Type 0 Header. It is permitted for VI to present an SDI with a different Type 0 Header (e.g., Vendor ID, Device ID, Class Code, Subsystem Vendor ID) and PCI Express Capabilities than its associated PF.↑

- ↑↑↑The device-specific extension to SR-PCIM provides a list of pages of memory (in the sizes implied by the System Page Size register) that make up each virtual BAR, and how these map onto the hardware or onto the extension itself.↑

↑↑↑When the SDI is presented as a Function, an SI interacts with it as if it had the BARs of a Function with a Type 0 Header. To do this, each page of memory is mapped onto a page of the PF BAR or the page is marked invalid in the page tables. When pages are invalid, the processor traps on access to them. This trapped access can be redirected to software, that supplies the values which would be read or written to the virtual BAR. In general, this process is known as either “emulation” or “full virtualization,” though the second term sometimes only applies to emulation of an entire device. By emulating some pages of a virtual BAR in software, a device designer can avoid replicating the setup and teardown parts of the hardware control surface into all of the SDIs.↑

- ↑↑↑If MSI-X is employed, the device-specific extension provides for emulating the MSI-X table.↑

 ECN: Base 6.3
 SIOV△↔

9.3.1.1.1.2 ↑↑ Requirements of an SI ↑ §

↑↑↑Any PCI Express Endpoint that supports TDISP may depend on an SI for specific functionality. The device drivers within the SIs optionally move the TDIs that the Endpoint supports from the CONFIG_UNLOCKED state into the CONFIG_LOCKED state, followed by a query for a DEVICE_INTERFACE_REPORT. If the fields in the DEVICE_INTERFACE_REPORT contain acceptable values for a secure configuration, the drivers move the TDI into the RUN state, which allows the SI to bring the TDI into the SI's Trusted Computing Base.↑

↑↑↑For an Endpoint that supports both TDISP and Scalable IOV, an SI accounts for the parts of the device interface that are implemented in SR-PCIM, including Configuration Space and any part of the initialization that isn't implemented in the hardware interface of the SDI. Since SR-PCIM may sit outside of the Trusted Computing Base, the device driver initializes the device interface before moving from CONFIG_UNLOCKED to CONFIG_LOCKED. Then, once the TDI is locked, the device interface report can be evaluated and then the TDI is moved to the RUN state and interactions with the SDI can be trusted.↑

- ↑↑↑An SI allows a device driver for an SDI within the SI to optionally interact with that SDI before moving the TDI from CONFIG_UNLOCKED to CONFIG_LOCKED.↑
- ↑↑↑An SI allows a device driver for an SDI to evaluate the DEVICE_INTERFACE_REPORT before moving a TDI from CONFIG_LOCKED to RUN.↑

9.3.1.2 ↑↑ MSI/MSI-X Interrupt Message Resource Allocation ↑ §

↑↑↑An MSI/MSI-X interrupt message is a Memory Write Request with system-specific values that differentiate it from other DMA Memory Write Requests. An interrupt message resource is the collection of system-specific values, such as address and data, needed to create the Memory Write Request. This specification puts no meaning on the specific values

or how they are assigned/derived for interrupt message resources. Like any other Memory Write Request from an SDI, this memory write must use the RID of the SDI.

↑↑Unlike BAR space resources, interrupt message resources are not required to be assigned initially to the PF via an architected mechanism (MSI or MSI-X) and subsequently delegated to an SDI, though an implementation is permitted to choose such an approach. In fact, SDIs have no requirement to store interrupt message resource values tuples (address/data/etc.) in a processor-visible address space like Configuration Space (as MSI does) or Memory Space (like MSI-X does). If they are stored in Memory Space, there is no requirement that they be stored contiguously or in special address ranges that can be isolated (like MSI-X requires).

↑↑The VI is permitted to present an SDI to an SI as a Function with a Type 0 Header, virtualizing the parts of the device interface that are not implemented in device hardware or firmware. If the VI presents an SDI as a Function with a Type 0 Header, it must present MSI Capability or MSI-X Capability structures and intercept any access to interrupt configuration within the SI, redirecting it to SR-PCIM. If the VI does not present an SDI as a Function with a Type 0 Header, SR-PCIM must provide alternative software interfaces supplying this functionality. SR-PCIM is free to use any mechanism the device exposes.

- ↑↑A PF with an SIOV Extended Capability is permitted to expose functionality isomorphic with MSI on each SDI, in which case it must store an address value for each SDI, a base data value for each SDI, and an index for each interrupt in use by the SDI, to be added to the base data value when generating an interrupt message.
- ↑↑A PF with an SIOV Extended Capability is permitted to expose functionality isomorphic with MSI-X on each SDI, in which case it must store interrupt message resource values (address/data/etc.) for each interrupt in use by the SDI.

9.4 ↑↑SR-IOV↑ Configuration §

9.4.1 SR-IOV Configuration Overview §

This section provides SR-IOV-added requirements for implementing PFs and VFs.

PFs are discoverable in configuration space, as with all Functions. PFs contain the SR-IOV Extended Capability described in § Section 9.4.3 . PFs are used to discover, configure, and manage the VFs associated with the PF and for other things described in this specification.

9.4.2 ↑↑SR-IOV↑ Configuration Space §

PFs that support SR-IOV shall implement the SR-IOV Extended Capability as defined in the following sections. VFs shall implement configuration space fields and capabilities as defined in the following sections.

9.4.3 SR-IOV Extended Capability §

The SR-IOV Extended Capability defined here is a PCIe extended capability that must be implemented in each PF that supports SR-IOV. This Capability is used to describe and control a PF's SR-IOV Capabilities.

For Multi-Function Devices , each PF that supports SR-IOV shall provide the Capability structure defined in this section. This Capability structure may be present in any Function with a Type 0 Configuration Space Header . This Capability must not be present in Functions with a Type 1 Configuration Space Header .

§ Figure 9-11 shows the SR-IOV Extended Capability structure.

Base 6.4 vs Base 6.3

| Byte Offset | |
|-------------|---|
| +000h | PCI Express Extended Capability Header |
| +004h | SR-IOV Capabilities Register (RO) |
| +008h | SR-IOV Status Register SR-IOV Control Register (RW) |
| +00Ch | TotalVFs (RO) InitialVFs (RO) |
| +010h | RsvdP Fcn Dep Link (RO) NumVFs (RW) |
| +014h | VF Stride (RO) FirstVF Offset (RO) |
| +018h | VF Device ID (RO) RsvdP |
| +01Ch | Supported Page Sizes (RO) |
| +020h | System Page Size (RW) |
| +024h | VF BAR0 (RW) |
| +028h | VF BAR1 (RW) |
| +02Ch | VF BAR2 (RW) |
| +030h | VF BAR3 (RW) |
| +034h | VF BAR4 (RW) |
| +038h | VF BAR5 (RW) |
| +03Ch | VF Migration State Array Offset (RO) |

Figure 9-11 SR-IOV Extended Capability §

9.4.3.1 SR-IOV Extended Capability Header (Offset 00h) §

§ Table 9-2 defines the SR-IOV Extended Capability header . The Capability ID for the SR-IOV Extended Capability is 0010h

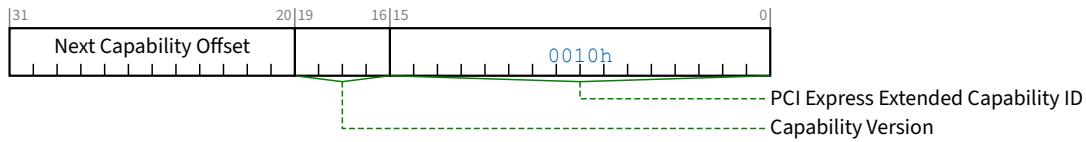


Figure 9-12 SR-IOV Extended Capability Header §

Table 9-2 SR-IOV Extended Capability Header §

| Bit Location ↑↓↑↓↑↑ | Register Description | Attributes |
|------------------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the SR-IOV Extended Capability is 0010h . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh. | RO |

9.4.3.2 SR-IOV Capabilities Register (04h) §

§ Table 9-3 defines the layout of the SR-IOV Capabilities field.

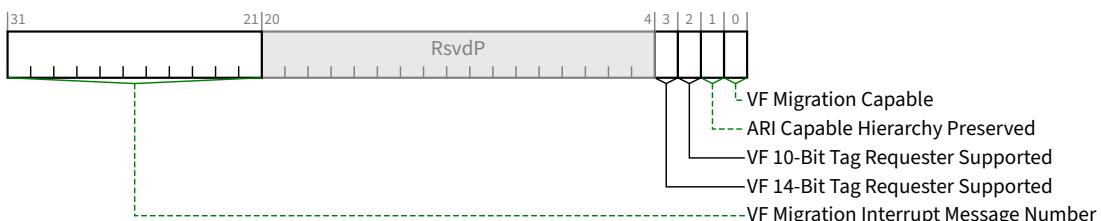


Figure 9-13 SR-IOV Capabilities Register §

Table 9-3 SR-IOV Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | VF Migration Capable - If Set, the PF is Migration Capable and operating under a Migration Capable MR-PCIM. Deprecated. New designs should hardwire this bit to 0b. | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|--|---------------------------------------|
| 1 | <p><i>ARI Capable Hierarchy Preserved</i></p> <p><i>PCI Express Endpoint:</i> If Set, the <u>ARI Capable Hierarchy</u> bit is preserved across certain power state transitions.</p> <p><i>RCiEP:</i> Not applicable - this bit <i>MUST</i>@FLIT be hardwired to 0b.</p> | RO |
| 2 | <p><i>VF 10-Bit Tag Requester Supported</i> - If Set, all VFs associated with this PF must support 10-Bit Tag Requester ↓↓capability↓↓capability for non-UIO Requests↑↑. If Clear, VFs must not support 10-Bit Tag Requester ↓↓capability↓↓capability for non-UIO Requests. This bit is not applicable to a Requester when generating UIO Requests↑↑</p> <p>If the <u>10-Bit Tag Requester Supported</u> bit in the PF's Device Capabilities 2 register is Clear, this bit must be Clear.</p> | HwInit Errata: Base 6.3 B807△◀▶ |
| 3 | <p><i>VF 14-Bit Tag Requester Supported</i> - If Set, all VFs associated with this PF must support 14-Bit Tag Requester ↓↓capability↓↓capability for non-UIO Requests↑↑. If Clear, VFs must not support 14-Bit Tag Requester ↓↓capability↓↓capability for non-UIO Requests. This bit is not applicable to a Requester when generating UIO Requests↑↑</p> <p>If the <u>14-Bit Tag Requester Supported</u> bit in the PF's Device Capabilities 3 register is Clear, this bit must be Clear.</p> | HwInit Errata: Base 6.3 B807△◀▶ |
| 31:21 | <p><i>VF Migration Interrupt Message Number</i> - Indicates the MSI/MSI-X vector used for migration interrupts. The value in this field is undefined if <u>VF Migration Capable</u> is Clear.</p> <p>VF Migration is a feature associated with the deprecated [MR-IOV]. This bit should be hardwired to 0b in new designs</p> | RO |

9.4.3.2.1 VF Migration Capable §

VF Migration Capable is Set to indicate that the PF supports VF Migration. If Clear, the PF does not support VF Migration.

VF Migration is a feature associated with the deprecated [MR-IOV]. This bit should be hardwired to 0b in new designs.

9.4.3.2.2 ARI Capable Hierarchy Preserved §

ARI Capable Hierarchy Preserved is Set to indicate that the PF preserves the ARI Capable Hierarchy bit across certain power state transitions (see § Section 9.4.3.3.5). Components must either Set this bit or Set the No_Soft_Reset bit (see § Section 5.10.2). It is recommended that components set this bit even if they also set No_Soft_Reset.

ARI Capable Hierarchy Preserved is only present in the lowest-numbered PF of a Device (for example PF0). ARI Capable Hierarchy Preserved is Read Only Zero in other PFs of a Device.

ARI Capable Hierarchy Preserved does not apply to RCiEPs, and its value is undefined (see § Section 9.4.3.3).

9.4.3.2.3 VF Larger-Tag Requester Support §

If a PF and/or its associated VFs support UIO as a Completer, 14-bit Tags must be supported.

Otherwise, if a PF supports one or both larger-Tag Requester capabilities, its associated VFs are permitted to support the associated larger-Tag Requester capabilities as well, but this is optional. Especially for usage models where the bulk of the traffic is spread across several VFs concurrently, it may not be necessary for individual VFs to use larger Tags so they can support >256 outstanding Non-Posted Requests each.

For a given PF, it is required that either all or none of its associated VFs support larger-Tag Requester capabilities. This avoids unnecessary implementation and management complexity. See [VF 14-Bit Tag Requester Supported](#) and [VF 10-Bit Tag Requester Supported](#).

VFs that support larger-Tag Requester capabilities have additional requirements and recommendations beyond other Function types in order to simplify error handling and reduce the possibility of larger-Tag related errors with one VF impacting other traffic.

- If one of the ~~↓↑VF larger Tag Requester Enable bits~~ ↑↓VF larger-Tag Requester Enable bits in the SR-IOV Control Register is Set, then each VF must use the associated larger Tags for all Non-Posted Requests that it generates. If both bits are Set at the same time, the result is undefined.
- As documented in § [Table 2-11](#), the permitted range with 10-bit Tags enabled is 256 to 1023, and the recommended range with 14-bit Tags enabled is 1024 to 16383. The keep-out range for 10-bit Tags is Tag[9:8] being 00b, and the recommended keep-out range for 14-bit Tags is Tag[13:10] being 0000b.
- For each outstanding ~~↓↑larger Tag Request~~, ↑↓larger-Tag Request, if the VF receives a Completion that matches the outstanding Request other than all Tag bits in the implemented keep-out Tag bit range being zero, then the Tag is invalid, and it is strongly recommended that the VF prevent that Request from (eventually) generating a Completion Timeout error, and instead handle the error via a device-specific mechanism that avoids data corruption.

It is strongly recommended that software not configure Unexpected Completion errors to be handled as Uncorrectable Errors. This avoids them triggering System Errors or hardware error containment mechanisms like Downstream Port Containment (DPC).

IMPLEMENTATION NOTE: NO VF LARGER-TAG COMPLETER SUPPORTED BITS §

There are no VF 14-bit or VF 10-Bit Tag Completer Supported bits. If a PF supports one or both larger-Tag Completer capabilities, then all of its associated VFs are required support the associated Tag Completer capabilities as stated in [14-Bit Tag Completer Supported](#) and [10-Bit Tag Completer Supported](#). This helps avoid the complexity of PCIe hierarchies where some Completers support larger-Tag capability and some do not.

9.4.3.2.4 VF Migration Interrupt Message Number §

VF Migration is associated with the now deprecated MR-IOV. New designs should hardwire this field to 0.

9.4.3.3 SR-IOV Control Register (Offset 08h) §

§ [Table 9-4](#) defines the layout of the SR-IOV Control fields.

Base 6.4 vs Base 6.3

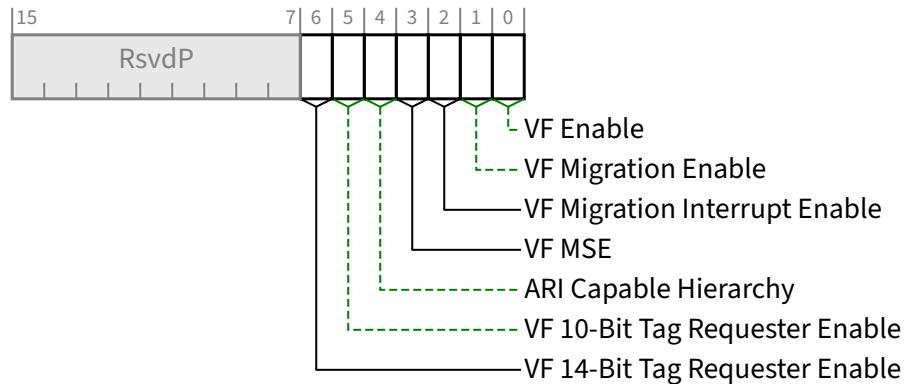


Figure 9-14 SR-IOV Control Register §

Table 9-4 SR-IOV Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|-------------------------------|
| 0 | VF Enable - Enables/Disables VFs. Default value is 0b. | RW |
| 1 | VF Migration Enable - Enables/Disables VF Migration Support. Default value is 0b. See § Section 9.4.3.3.2 . | RW or RO (see description) |
| 2 | VF Migration Interrupt Enable - Enables/Disables VF Migration State Change Interrupt. Default value is 0b. | RW |
| 3 | VF MSE - Memory Space Enable for Virtual Functions . Default value is 0b. | RW |
| 4 | ARI Capable Hierarchy - <i>PCI Express Endpoint:</i> This bit must be RW in the lowest-numbered PF of the Device and hardwired to 0b in all other PFs. If the value of this bit is 1b, the Device is permitted to locate VFs in Function Numbers 8 to 255 of the captured Bus Number. Otherwise, the Device must locate VFs as if it were a non-ARI Device. This bit is not affected by FLR of any PF or VF. Default value is 0b. <i>RCiEP:</i> Not applicable - this bit must be hardwired to 0b. Within the Root Complex, VFs are always permitted to be assigned to any Function Number allowed by First VF Offset and VF Stride rules (see § Section 9.4.3.9 and § Section 9.4.3.10). VF 10-Bit Tag Requester Enable - If Set, all VFs must use 10-Bit Tags for all ↑↑non-UIO,↑ Non-Posted Requests they generate. If Clear, VFs must not use 10-Bit Tags for ↑↑non-UIO,↑ Non-Posted Requests they generate. ↑↑This bit is not applicable to a Requester when generating UIO Requests.↑ See VF Larger-Tag Requester Support . This bit must not be Set if the VF 14-Bit Tag Requester Enable bit is Set; otherwise the result is undefined. If software changes the value of this bit while any VFs have outstanding Non-Posted Requests, the result is undefined. | RW or RO (see description) |
| 5 | | RW or RO |

Errata: Base 6.3 B807△<

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | If the VF 10-Bit Tag Requester Supported bit in the SR-IOV Capabilities register is Clear, this bit is permitted to be hardwired to 0b. Default value is 0b. | |
| 6 | VF 14-Bit Tag Requester Enable - If Set, all VFs must use 14-Bit Tags for all Non-Posted Requests they generate. If Clear, VFs must not use 14-Bit Tags for Non-Posted Requests they generate. See VF Larger-Tag Requester Support . This bit must not be Set if the VF 10-Bit Tag Requester Enable bit is Set; otherwise the result is undefined. If software changes the value of this bit while any VFs have outstanding Non-Posted Requests, the result is undefined. If the VF 14-Bit Tag Requester Supported bit in the SR-IOV Capabilities register is Clear, this bit is permitted to be hardwired to 0b. Default value is 0b. | RW or RO |

9.4.3.3.1 VF Enable §

VF Enable manages the assignment of VFs to the associated PF. If VF Enable is Set, the VFs associated with the PF are accessible in the PCI Express fabric. When Set, VFs respond to and may issue PCI Express transactions following the rules for PCI Express Endpoint Functions.

If VF Enable is Clear, VFs are disabled and not visible in the PCI Express fabric; requests to these VFs shall receive UR and these VFs shall not issue PCI Express transactions.

To allow components to perform internal initialization, after changing the VF Enable bit from Cleared to Set, the system is not permitted to issue Requests to the VFs which are enabled by that VF Enable bit until one of the following is true:

- At least 100 ms has passed
- An FRS Message has been received from the PF with a Reason Code of VF Enable d
- At least VF Enable Time has passed. VF Enable Time is either (1) the Reset Time value in the Readiness Time Reporting Extended Capability associated with the VF, or (2) a value determined by system software / firmware ²⁰⁷.

The Root Complex and/or system software must allow at least 1.0 s after Setting the VF Enable bit, before it may determine that a VF which fails to return a Successful Completion Status for a valid Configuration Request is broken. After Setting the VF Enable bit, the VFs enabled by that VF Enable bit are permitted to return a Configuration RRS status in response to Configuration Requests up to the 1.0 s limit, if they are not ready to provide a Successful Completion Status for a valid Configuration Request. After a PF transmits an FRS Message with a Reason Code of VF Enable d, no VF associated with that PF is permitted to return Configuration RRS in response to a Configuration Request without an intervening VF disable or other valid reset condition. After returning a Successful Completion to any Request, no VF is permitted to return Configuration RRS in response to a Configuration Request without an intervening VF disable or other valid reset condition.

Since VFs don't have an MSE bit (MSE in VFs is controlled by the VF MSE bit in the SR-IOV Extended Capability in the PF), it's possible for software to issue a Memory Request before the VF is ready to handle it. Therefore, Memory Requests must not be issued to a VF until at least one of the conditions listed in § Section 5.10.1 regarding the VF's internal initialization completing has been met.

²⁰⁷. For example, ACPI tables.

The VF is permitted to silently drop Memory Requests after an FLR has been issued to the VF or VF Enable has been Set in the associated PF's SR-IOV Extended Capability until the VF responds successfully to any Request (excluding the return of RRS in response to a Configuration Request).

Clearing VF Enable effectively destroys the VFs. Setting VF Enable effectively creates VFs. Setting VF Enable after it has previously been Cleared shall result in a new set of VFs. If the PF is in the D0 power state, the new VFs are in the D0 uninitialized state. If the PF is in a lower power state behavior is undefined (see Sections 9.6.1 and 9.6.2).

When Clearing VF Enable , a PF that supports FRS shall send an FRS Message with FRS Reason VF Disabled to indicate when this operation is complete. The PF is not permitted to send this Message if there are outstanding Non-Posted ^{↑↑or}
UIO[↑] Requests issued by the PF or any of the VFs associated with the PF. The FRS Message may only be sent after these Requests have completed (or timed out).

Errata: Base 6.3
B834△◀▷

After VF Enable is Cleared no field in the SR-IOV Extended Capability may be accessed until either:

- At least 1.0 s has elapsed after VF Enable was Cleared.
- The PF supports FRS and after VF Enable was Cleared, an FRS Message has been received from the PF with a Reason Code of VF Disabled.

^{↑↑If the PF has both an SR-IOV Extended Capability and an SIOV Extended Capability , setting VF Enable while SIOV Enabled is set is prohibited. In other words, SR-IOV and SIOV may not be used simultaneously in the PF.↑}

ECN: Base 6.3
SIOV△◀▷

§ Section 9.4.3.7 NumVFs , § Section 9.4.3.5 InitialVFs , § Section 9.4.3.6 TotalVFs , § Section 9.4.3.9 First VF Offset , § Section 9.4.3.13 System Page Size , and § Section 9.4.3.14 VF BARx describe additional semantics associated with this field.

9.4.3.3.2 VF Migration Enable [§]

VF Migration Enable must be Set to allow VF Migration on this PF.

VF Migration is associated with the now-deprecated [MR-IOV]. New designs should hardwire this bit to 0b.

9.4.3.3.3 VF Migration Interrupt Enable [§]

VF Migration is associated with the now-deprecated MR-IOV. New designs should hardwire this bit to 0b.

9.4.3.3.4 VF MSE (Memory Space Enable) [§]

VF MSE controls memory space enable for all Active VFs associated with this PF, as with the Memory Space Enable bit in a Function's PCI Command register. The default value for this bit is 0b.

When VF Enable is Set, VF memory space will respond only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the [PCIe] if an attempt is made to access a Virtual Function's memory space when VF Enable is Set and VF MSE is Clear.

IMPLEMENTATION NOTE: VF MSE AND VF ENABLE §

VF memory space will respond with Unsupported Request when VF Enable is Clear. Thus, VF MSE is “don’t care” when VF Enable is Clear; however, software may choose to Set VF MSE after programming the VF BARn registers, prior to Setting VF Enable .

9.4.3.3.5 ARI Capable Hierarchy §

For Devices associated with an Upstream Port, ARI Capable Hierarchy is a hint to the Device that ARI has been enabled in the Root Port or Switch Downstream Port immediately above the Device. Software should set this bit to match the ARI Forwarding Enable bit in the Root Port or Switch Downstream Port immediately above the Device.

ARI Capable Hierarchy is only present in the lowest-numbered PF of a Device (for example PF₀) and affects all PFs of the Device. ARI Capable Hierarchy is Read Only Zero in other PFs of a Device.

A Device may use the setting of ARI Capable Hierarchy to determine the values for First VF Offset (see § Section 9.4.3.9) and VF Stride (see § Section 9.4.3.10). The effect of changing ARI Capable Hierarchy is undefined if VF Enable is Set in any PF. This bit must be set to its default value upon Conventional Reset. This bit is not affected by FLR of any PF or VF. If either ARI Capable Hierarchy Preserved is Set (see § Section 9.4.3.2.2) or No_Soft_Reset is Set (see § Section 5.10.2), a power state transition of this PF from D3_{Hot} to D0 does not affect the value of this bit (see § Section 5.10.2).

ARI Capable Hierarchy does not apply to RCiEPs.

IMPLEMENTATION NOTE: ARI CAPABLE HIERARCHY §

For a Device associated with an Upstream Port, that Device has no way of knowing whether ARI has been enabled. If ARI is enabled, the Device can conserve Bus Numbers by assigning VFs to Function Numbers greater than 7 on the captured Bus Number. ARI is defined in § Section 6.13.

Since RCiEPs are not associated with an Upstream Port, ARI does not apply, and VFs may be assigned to any Function Number within the Root Complex permitted by First VF Offset and VF Stride (see § Section 9.4.3.8 and § Section 9.4.3.9).

9.4.3.4 SR-IOV Status Register (Offset 0Ah) §

§ Table 9-5 : defines the layout of the SR-IOV Status field.

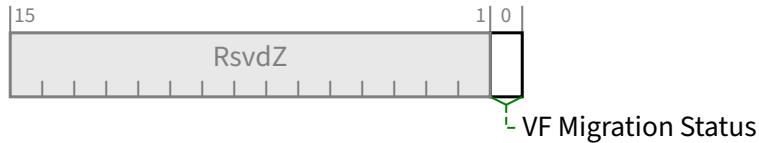


Figure 9-15 SR-IOV Status §

Table 9-5 SR-IOV Status §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | VF Migration Status - VF Migration is associated with the now-deprecated [MR-IOV]. New designs should hardwire this bit to 0b. | RW1C |

9.4.3.4.1 VF Migration Status §

VF Migration is associated with the now-deprecated [MR-IOV]. New designs should hardwire this bit to 0b.

9.4.3.5 InitialVFs (Offset 0Ch) §

VF Migration is associated with the now deprecated MR-IOV. New designs should encode this field as HwInit with a value equal to TotalVFs.

9.4.3.6 TotalVFs (Offset 0Eh) §

TotalVFs indicates the maximum number of VFs that are associated with the PF.

The minimum value of TotalVFs is 0.

This field is HwInit and must contain the same value as InitialVFs .

9.4.3.7 NumVFs (Offset 10h) §

NumVFs controls the number of VFs that are visible. SR-PCIM sets NumVFs as part of the process of creating VFs. This number of VFs shall be visible in the PCI Express fabric after both NumVFs is set to a valid value and VF Enable is Set.

The results are undefined if NumVFs is set to a value greater than TotalVFs .

NumVFs may only be written while VF Enable is Clear. If NumVFs is written when VF Enable is Set, the results are undefined.

The initial value of NumVFs is undefined.

9.4.3.8 Function Dependency Link (Offset 12h) §

The programming model for a Device may have vendor-specific dependencies between sets of Functions. The Function Dependency Link field is used to describe these dependencies.

This field describes dependencies between PFs. VF dependencies are the same as the dependencies of their associated PFs.

If a PF is independent from other PFs of a Device, this field shall contain its own Function Number.

If a PF is dependent on other PFs of a Device, this field shall contain the Function Number of the next PF in the same Function Dependency List. The last PF in a Function Dependency List shall contain the Function Number of the first PF in the Function Dependency List.

If PF_p and PF_q are in the same Function Dependency List, then any SI that is assigned $\text{VF}_{p,n}$ shall also be assigned to $\text{VF}_{q,n}$.

IMPLEMENTATION NOTE: FUNCTION DEPENDENCY LINK EXAMPLE §

Consider the following scenario:

| SR-IOV Field | PF 0 | PF 1 | PF 2 |
|--------------------------|------|------|------|
| Function Dependency Link | 1 | 0 | 2 |
| NumVFs | 4 | 4 | 6 |
| First VF Offset | 4 | 4 | 4 |
| VF Stride | 3 | 3 | 3 |

| Function Number | Description | Independent |
|-----------------|------------------------|-------------|
| 0 | PF 0 | No |
| 1 | PF 1 | No |
| 2 | PF 2 | Yes |
| 3 | Function not present | |
| 4 | VF 0,1 (aka PF 0 VF 1) | No |
| 5 | VF 1,1 (aka PF 1 VF 1) | No |
| 6 | VF 2,1 (aka PF 2 VF 1) | Yes |
| 7 | VF 0,2 | No |
| 8 | VF 1,2 | No |
| 9 | VF 2,2 | Yes |
| 10 | VF 0,3 | No |
| 11 | VF 1,3 | No |
| 12 | VF 2,3 | Yes |
| 13 | VF 0,4 | No |
| 14 | VF 1,4 | No |
| 15 | VF 2,4 | Yes |
| 16 to 17 | Functions not present | |
| 18 | VF 2,5 | Yes |
| 19 to 20 | Functions not present | |
| 21 | VF 2,6 | Yes |

| Function Number | Description | Independent |
|-----------------|-----------------------|-------------|
| 22 to 255 | Functions not present | |

In this example, Functions 4 and 5 must be assigned to the same SI. Similarly, Functions 7 and 8, 10 and 11, and 13 and 14 must be assigned together. If PFs are assigned to SIs, Functions 0 and 1 must be assigned together as well. Functions 2, 6, 9, 12, 15, 18, and 21 are independent and may be assigned to SIs in any fashion.

All PFs in a Function Dependency List shall have the same values for the InitialVFs and TotalVFs fields.

SR-PCIM shall ensure that all PFs in a Function Dependency List have the same values for the NumVFs and VF Enable fields before any VF in that Function Dependency List is assigned to an SI.

VF Mapping operations occur independently for every VF. SR-PCIM shall not assign a VF to an SI until it can assign all dependent VFs.

Similarly, SR-PCIM shall not remove a VF from an SI until it can remove all dependent VFs.

9.4.3.9 First VF Offset (Offset 14h) §

First VF Offset is a constant and defines the Routing ID offset of the first VF that is associated with the PF that contains this Capability structure. The first VF's 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the PF containing this field ignoring any carry, using unsigned, 16-bit arithmetic.

A VF shall not be located on a Bus Number that is numerically smaller than its associated PF.

This field may change value when the lowest-numbered PF's ARI Capable Hierarchy value changes or when this PF's NumVFs value changes.

Note: First VF Offset is unused if NumVFs is 0. If NumVFs is greater than 0, First VF Offset must not be zero.

9.4.3.10 VF Stride (Offset 16h) §

VF Stride defines the Routing ID offset from one VF to the next one for all VFs associated with the PF that contains this Capability structure. The next VF's 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the current VF, ignoring any carry, using unsigned 16-bit arithmetic.

This field may change value when the lowest-numbered PF's ARI Capable Hierarchy value changes or when this PF's NumVFs value changes.

Note: VF Stride is unused if NumVFs is 0 or 1. If NumVFs is greater than 1, VF Stride must not be zero.

9.4.3.11 VF Device ID (Offset 1Ah) §

This field contains the Device ID that should be presented for every VF to the SI.

VF Device ID may be different from the PF Device ID. A VF Device ID must be managed by the vendor. The vendor must ensure that the chosen VF Device ID does not result in the use of an incompatible device driver.

9.4.3.12 Supported Page Sizes (Offset 1Ch) §

This field indicates the page sizes supported by the PF. This PF supports a page size of 2^{n+12} if bit n is Set. For example, if bit 0 is Set, the PF supports 4-KB page sizes. PFs are required to support 4-KB, 8-KB, 64-KB, 256-KB, 1-MB, and 4-MB page sizes. All other page sizes are optional.

The page size describes the minimum alignment requirements for VF BAR resources as described in § [Section 9.4.3.13](#).

9.4.3.13 System Page Size (Offset 20h) §

This field defines the page size the system will use to map the VFs' memory addresses. Software must set the value of the System Page Size to one of the page sizes set in the Supported Page Sizes field (see § [Section 9.4.3.12](#)). As with Supported Page Sizes, if bit n is Set in System Page Size, the VFs associated with this PF are required to support a page size of 2^{n+12} . For example, if bit 1 is Set, the system is using an 8-KB page size. The results are undefined if System Page Size is zero. The results are undefined if more than one bit is Set in System Page Size. The results are undefined if a bit is Set in System Page Size that is not Set in Supported Page Sizes.

When System Page Size is set, the VF associated with this PF is required to align all BAR resources on a System Page Size boundary. Each VF BAR n or VF BAR n pair (see § [Section 9.4.3.14](#)) shall be aligned on a System Page Size boundary. Each VF BAR n or VF BAR n pair defining a non-zero address space shall be sized to consume an integer multiple of System Page Size bytes. All data structures requiring page size alignment within a VF shall be aligned on a System Page Size boundary.

VF Enable must be zero when System Page Size is written. The results are undefined if System Page Size is written when VF Enable is Set.

Default value is 0000 0001h (i.e., 4 KB).

9.4.3.14 VF BAR0 (Offset 24h), VF BAR1 (Offset 28h), VF BAR2 (Offset 2Ch), VF BAR3 (Offset 30h), VF BAR4 (Offset 34h), VF BAR5 (Offset 38h) §

These fields must define the VF's Base Address Registers (BARs). These fields behave as normal PCI BARs, as described in § [Section 7.5.1](#). They can be sized by writing all 1s and reading back the contents of the BARs as described in § [Section 7.5.1.2.1](#), complying with the low order bits that define the BAR type fields.

These fields may have their attributes affected by the VF Resizable BAR Extended Capability (see § [Section 7.8.7](#)) if it is implemented.

The amount of address space decoded by each BAR shall be an integral multiple of System Page Size.

Each VF BAR n , when "sized" by writing 1s and reading back the contents, describes the amount of address space consumed and alignment required by a single Virtual Function, per BAR. When written with an actual address value, and VF Enable and VF MSE are Set, the BAR maps NumVFs BARs. In other words, the base address is the address of the first VF BAR n associated with this PF and all subsequent VF BAR n address ranges follow as described below.

VF BARs shall only support 32-bit and 64-bit memory space. PCI I/O Space is not supported in VFs. Bit 0 of any implemented VF BARx must be RO 0b except for a VF BARx used to map the upper 32 bits of a 64-bit memory VF BAR pair.

The alignment requirement and size read is for a single VF, but when VF Enable is Set and VF MSE is Set, the BAR contains the base address for all (NumVFs) VF BAR n .

The algorithm to determine the amount of address space mapped by a VF BAR n differs from the standard BAR algorithm as follows:

1. Resize the BAR via the VF Resizable BAR Extended Capability (see § [Section 7.8.7](#)) if it is implemented.
2. After reading the low order bits to determine if the BAR is a 32-bit BAR or 64-bit BAR pair, determine the size and alignment requirements by writing all 1s to VF BAR n (or VF BAR n and VF BAR $n+1$ for a 64-bit BAR pair) and reading back the contents of the BAR or BAR pair. Convert the bit mask returned by the read(s) to a size and alignment value as described in § [Section 7.5.1.2.1](#). This value is the size and alignment for a single VF.
3. Multiply the value from step 2 by the value set in [NumVFs](#) to determine the total amount of space the BAR or BAR pair will map after [VF Enable](#) and [VF MSE](#) are Set.

For each VF BAR n field, n corresponds to one of the VFs BAR spaces. § [Table 9-8](#) shows the relationship between n and a Function's BAR.

Table 9-8 BAR Offsets §

| n | BAR Offset in a Type 0 Header |
|-----|-------------------------------|
| 0 | 10h |
| 1 | 14h |
| 2 | 18h |
| 3 | 1Ch |
| 4 | 20h |
| 5 | 24h |

The contents of all VF BAR n registers are indeterminate after [System Page Size](#) is changed.

9.4.3.15 VF Migration State Array Offset (Deprecated) (Offset 3Ch) §

VF Migration is associated with the now deprecated [MR-IOV]. New designs should hardwire this register to 0000 0000h.

9.4.4 PF/VF Configuration Space Header §

This section's material in previous versions of this specification has been integrated into § [Section 7.5.1](#).

9.4.5 PCI Express Capability Changes §

This section's material in previous versions of this specification has been integrated into § [Section 7.5.3](#).

9.4.6 PCI Standard Capabilities §

SR-IOV usage of PCI Standard Capabilities is described in § [Table 9-9](#). Items marked n/a are not applicable to PFs or VFs.

Base 6.4 vs Base 6.3

Table 9-9 SR-IOV Usage of PCI Standard Capabilities §

| Capability ID | Description | PF Attributes | VF Attributes |
|---------------|-------------------------------------|---|--|
| 00h | Null Capability | Base | Base |
| 01h | PCI Power Management Interface | Base | Optional. See § Section 5.10 . |
| 02h | AGP | n/a | n/a |
| 03h | VPD | Base | Optional. See § Section 7.9.18 . |
| 04h | Slot Identification | n/a | n/a |
| 05h | MSI | Base | See § Section 9.2.1.4 . |
| 06h | CompactPCI Hot Swap | n/a | n/a |
| 07h | PCI-X | n/a | n/a |
| 08h | HyperTransport | n/a | n/a |
| 09h | Vendor-specific | Base | Base |
| 0Ah | Debug Port | Base | Base |
| 0Bh | CompactPCI Central Resource Control | n/a | n/a |
| 0Ch | PCI Hot Plug | Base | n/a |
| 0Dh | PCI Bridge Subsystem ID | n/a | n/a |
| 0Eh | AGP 8x | n/a | n/a |
| 0Fh | Secure Device | n/a | n/a |
| 10h | PCI Express | Base | See § Section 9.4.5 . |
| 11h | MSI-X | See § Section 9.2.1.4 . | See § Section 9.2.1.4 . |
| 12h | Serial ATA Data/Index Configuration | Base | n/a |
| 13h | Advanced Features | n/a | n/a |
| 14h | Enhanced Allocation | Base | Must not implement. |
| 15h | Flattening Portal Bridge (FPB) | n/a | n/a |

9.4.7 PCI Express Extended Capabilities Changes §

SR-IOV usage of PCI Express Extended Capabilities is described in § [Table 9-10](#). Items marked n/a are not applicable to PFs or VFs (e.g., for capabilities only present in Root Complexes, only present in Function 0, or only for managing a Port).

Table 9-10 SR-IOV Usage of PCI Express Extended Capabilities §

| Extended Capability ID | Description | PF Attributes | VF Attributes |
|------------------------|--|---------------|----------------------------|
| 0000h | Null Capability | Base | Base |
| 0001h | Advanced Error Reporting Extended Capability (AER) | Base | See capability description |

Base 6.4 vs Base 6.3

| Extended Capability ID | Description | PF Attributes | VF Attributes |
|------------------------|---|----------------------------|---|
| 0002h | <u>Virtual Channel Extended Capability (02h)</u> | Base | Must not implement. See capability description |
| 0003h | <u>Device Serial Number Extended Capability</u> | Base | See capability description |
| 0004h | <u>Power Budgeting Extended Capability</u> | Base | Must not implement. See capability description |
| 0005h | <u>Root Complex Link Declaration Extended Capability</u> | n/a | n/a |
| 0006h | <u>Root Complex Internal Link Control Extended Capability</u> | n/a | n/a |
| 0007h | <u>Root Complex Event Collector Endpoint Association Extended Capability</u> | n/a | n/a |
| 0008h | <u>Multi-Function Virtual Channel Extended Capability</u> | Base | n/a; only present in Function 0 |
| 0009h | <u>Virtual Channel Extended Capability (09h)</u> | Base | Must not implement. See capability description |
| 000Ah | <u>RCRB Header Extended Capability</u> | n/a | n/a |
| 000Bh | <u>Vendor-specific Extended Capability</u> | Base | Base |
| 000Ch | Deprecated; formerly used for the Configuration Access Correlation Extended Capability | n/a | n/a |
| 000Dh | <u>ACS Extended Capability</u> | See capability description | See capability description |
| 000Eh | <u>ARI Extended Capability (ARI)</u> | See capability description | See capability description |
| 000Fh | <u>ATS Extended Capability</u> | See capability description | See capability description |
| 0010h | <u>SR-IOV Extended Capability</u> | See capability description | Must not implement. See capability description |
| 0011h | Deprecated; formerly used for the MR-IOV Extended Capability (MR-IOV) | n/a | n/a |
| 0012h | <u>Multicast Extended Capability</u> | See capability description | See capability description |
| 0013h | <u>Page Request Extended Capability (PRI)</u> | See capability description | See capability description |
| 0014h | Reserved for AMD | Base | Base |

Base 6.4 vs Base 6.3

| Extended Capability ID | Description | PF Attributes | VF Attributes |
|------------------------|---|-------------------------------|---|
| 0015h | <u>Resizable BAR Extended Capability</u> | Base | Must not implement. See capability description |
| 0016h | <u>Dynamic Power Allocation Extended Capability (DPA)</u> | See capability description | Must not implement. See capability description |
| 0017h | <u>TPH Requester Extended Capability (TPH)</u> | See capability description | See capability description |
| 0018h | <u>LTR Extended Capability</u> | Base | n/a; only present in Function 0 |
| 0019h | <u>Secondary PCI Express Extended Capability</u> | Base | n/a; only present in Function 0 |
| 001Ah | <u>PMUX Extended Capability</u> | Base | n/a; only for managing a Port |
| 001Bh | <u>PASID Extended Capability</u> | Base | Must not implement |
| 001Ch | Deprecated; formerly used for the LN Requester Extended Capability | n/a | n/a |
| 001Dh | <u>DPC Extended Capability</u> | n/a; only for managing a Port | n/a; only for managing a Port |
| 001Eh | <u>L1 PM Substates Extended Capability</u> | Base | n/a; only present in Function 0 |
| 001Fh | <u>Precision Time Measurement Extended Capability (PTM)</u> | Base | n/a; only for managing a Port |
| 0020h | <u>PCI Express over M-PHY Extended Capability (M-PCIe)</u> | Base | n/a; only for managing a Port |
| 0021h | <u>FRS Queueing Extended Capability</u> | n/a | n/a |
| 0022h | <u>Readiness Time Reporting Extended Capability</u> | Base | See capability description |
| 0023h | <u>Designated vendor-specific Extended Capability</u> | Base | Base |
| 0024h | <u>VF Resizable BAR Extended Capability</u> | See capability description | Must not implement. See capability description |
| 0025h | <u>Data Link Feature Extended Capability</u> | Base | n/a; only for managing a Port |
| 0026h | <u>Physical Layer 16.0 GT/s Extended Capability</u> | Base | n/a; only for managing a Port |
| 0027h | <u>Lane Margining at the Receiver Extended Capability</u> | Base | n/a; only for managing a Port |
| 0028h | <u>Hierarchy ID Extended Capability</u> | Base | Base |

Base 6.4 vs Base 6.3

| Extended Capability ID | Description | PF Attributes | VF Attributes |
|-------------------------------|--|-------------------------------|--|
| 0029h | <u>Native PCIe Enclosure Management Extended Capability (NPEM)</u> | Base | n/a; only for managing a Port |
| 002Ah | <u>Physical Layer 32.0 GT/s Extended Capability</u> | Base | Must not implement |
| 002Bh | <u>Alternate Protocol Extended Capability</u> | Base | Must not implement |
| 002Ch | <u>SFI Extended Capability (System Firmware Intermediary)</u> | Base | Must not implement |
| 002Dh | <u>Shadow Functions Extended Capability</u> | Base | Base |
| 002Eh | <u>Data Object Exchange Extended Capability</u> | Base | Base |
| 002Fh | <u>Device 3 Extended Capability</u> | Base | Base |
| 0030h | <u>IDE Extended Capability</u> | Base | n/a; only present in Function 0 |
| 0031h | <u>Physical Layer 64.0 GT/s Extended Capability</u> | Base | Must not implement |
| 0032h | <u>Flit Logging Extended Capability</u> | Base | Must not implement |
| 0033h | <u>Flit Performance Measurement Extended Capability</u> | Base | Must not implement |
| 0034h | <u>Flit Error Injection Extended Capability</u> | Base | Must not implement |
| 0035h | <u>Streamlined Virtual Channel Extended Capability (SVC)</u> | Base | Must not implement. See capability description |
| 0036h | <u>MMIO Register Block Locator Extended Capability (MRBL)</u> | Base | Base |
| 0037h | <u>NOP Flit Extended Capability</u> | Base | Must not implement |
| ↑↑0038h↑ ECN: Base 6.3 SIOV△↔ | ↑↑SIOV Extended Capability↑ | ↑↑See capability description↑ | ↑↑Must not implement↑ ↑↑See capability description↑ |

9.5 ↑↑SIOV Configuration↑↑ §

9.5.1 ↑↑SIOV Configuration Overview↑↑ §

ECN: Base 6.3
SIOV△↔

↑↑This section provides SIOV-added requirements for implementing PFs.↑↑

↑↑PFs are discoverable in configuration space, as with all Functions. PFs contain the SIOV Extended Capability.↑↑

9.5.2 SIOV Configuration Space

↑↑PFs that support SIOV shall implement the SIOV Extended Capability as defined in the following sections. Scalable Device Interfaces (SDIs) implement no Configuration Space.↑

9.5.3 ↑↑SIOV Extended Capability↑ §

↑↑The SIOV Extended Capability defined here is a PCIe Extended Capability that must be implemented in each PF that supports SIOV. This Capability is used to describe and control a PF's SIOV Capabilities.↑

↑↑For Multi-Function Devices, each PF that supports SIOV shall provide the capability structure defined in this section. This capability structure may be present in any Function with a Type 0 Configuration Space Header. This Capability must not be present in Functions with a Type 1 Configuration Space Header.↑

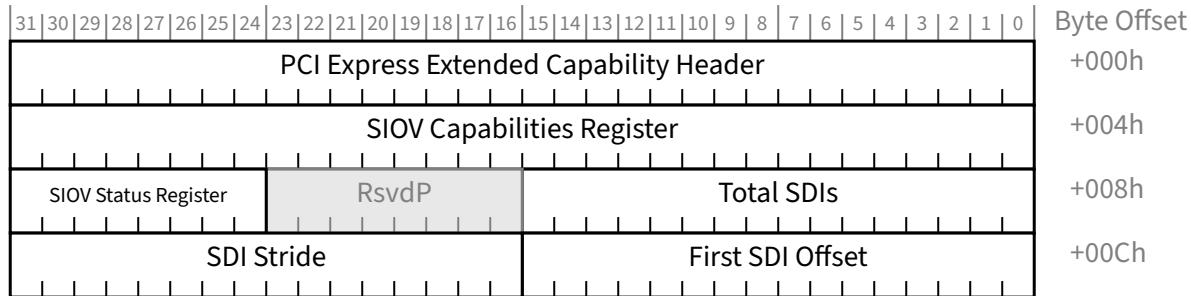


Figure 9-16 ↑↑SIOV Extended Capability↑ §

9.5.4 SIOV Extended Capability Header (Offset 00h)

↑↑§ Table 9-11 defines the SIOV Extended Capability header. The Capability ID for the SIOV Extended Capability is↑
↑↑0038h↑.

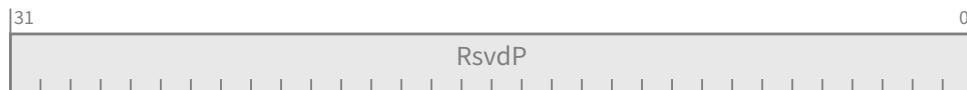


Figure 9-17 EditorialSIOV Extended Capability Header §

Table 9-11 ↑↑SIOV Extended Capability Header↑ ↑↑

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. | RO |

Base 6.4 vs Base 6.3

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| | The Extended Capability ID for the SIOV Extended Capability is 0038h. | |
| 19:16 | <p>Capability Version - This field is a PCI-SIG defined version number that indicates the version of the capability structure present.</p> <p>Must be 1h for this version of the specification.</p> | RO |
| 31:20 | <p>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.</p> <p>For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than OFFh.</p> | RO |

9.5.5 SIOV Capabilities Register (04h) §

§ Table 9-12 defines the layout of the SIOV Capabilities register.



Figure 9-18 Editorial SIOV Capabilities Register §

Table 9-12 SIOV Capabilities Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | <p>Strided RID Mapping – This PF uses the Strided RID Mapping. The only RID that may be assigned to an SDI is the one that is implied by First SDI Offset and SDI Stride. See § Section 9.1.1.</p> | RO |
| 1 | <p>Mixed IOV Supported – If no other PF exists in this device, or if no other PF supports SR-IOV, then this bit has no meaning and is hardwired to 0.</p> <p>If another PF in this device exists, and this bit is 0, then software that enables IOV must choose either SR-IOV (setting VF Enable) for all PFs within the device or choose SIOV (enabling SDIs) for all PFs. If this bit is 1, software may choose to enable SR-IOV on some PFs and SIOV on others.</p> | RO |

9.5.6 Total SDIs (Offset 08h) §

This field is RO and indicates the total number of SDIs supported by this Physical Function. Software can use this field to determine the total number of RIDs that this Physical Function would need for its SDIs, if all were simultaneously in use.

9.5.7 $\uparrow\downarrow$ SIOV Status Register (Offset 0Bh) \uparrow §



Figure 9-19 Editorial SIOV Status Register §

Table 9-13 $\uparrow\downarrow$ SIOV Status Register \uparrow §

| $\uparrow\downarrow$ Bit Location \uparrow | $\uparrow\downarrow$ Register Description \uparrow | $\uparrow\downarrow$ Attributes \uparrow |
|--|---|--|
| $\uparrow\downarrow$ 0 \uparrow | <p>$\uparrow\downarrow$SIOV Enabled – This bit indicates that at least one SDI is enabled (through device-specific means). This bit is Cleared when all SDIs are disabled. System software may use this bit to determine that a RID associated with an SDI is in active use. If system software finds this bit Set during system boot, or after system reset, a Function-Level Reset may be required to ensure that no SDI is active.\uparrow</p> <p>$\uparrow\downarrow$Default value of this bit is 0.\uparrow</p> | $\uparrow\downarrow$ RO \uparrow |

9.5.8 $\uparrow\downarrow$ First SDI Offset (Offset 0Ch) \uparrow §

$\uparrow\downarrow$ This field is RO and valid only when the Strided RID Mapping bit is Set in the SIOV Capabilities Register . If that bit is Clear, this field is RsvdP. \uparrow

$\uparrow\downarrow$ First SDI Offset is a constant and defines the RID offset of the first SDI that is associated with the PF that contains this capability structure. The first SDI's RID is calculated by adding the contents of this field to the RID of the PF containing this field ignoring any carry, using unsigned, 16-bit arithmetic. \uparrow

9.5.9 $\uparrow\downarrow$ SDI Stride (Offset 0Eh) \uparrow §

$\uparrow\downarrow$ This field is RO and valid only when the Strided RID Mapping bit is Set in the SIOV Capabilities Register . If that bit is Clear, this field is RsvdP. \uparrow

$\uparrow\downarrow$ SDI Stride defines the RID offset from one SDI to the next one for all SDIs associated with the PF that contains this capability structure. The next SDI's RID is calculated by adding the contents of this field to the RID of the current SDI, ignoring any carry, using unsigned 16-bit arithmetic. \uparrow

$\uparrow\downarrow$ If Strided RID Mapping is Set, SDI Stride must not be Zero. \uparrow

Base 6.4 vs Base 6.3

10. Address Translation Services (ATS) §

10.1 ATS Architectural Overview §

Most contemporary system architectures make provisions for translating addresses from DMA (bus mastering) I/O Functions. In many implementations, it has been common practice to assume that the physical address space seen by the CPU and by an I/O Function is equivalent. While in others, this is not the case. The address programmed into an I/O Function is a “handle” that is processed by the Root Complex (RC). The result of this processing is often a translation to a physical memory address within the central complex. Typically, the processing includes access rights checking to insure that the DMA Function is allowed to access the referenced memory location(s).

The purposes for having DMA address translation vary and include:

- Limiting the destructiveness of a “broken” or misprogrammed DMA I/O Function
- Providing for scatter/gather
- Ability to redirect message-signaled interrupts (e.g., MSI or MSI-X) to different address ranges without requiring coordination with the underlying I/O Function
- Address space conversion (32-bit I/O Function to larger system address space)
- Virtualization support

Irrespective of the motivation, the presence of DMA address translation in the host system has certain performance implications for DMA accesses.

Depending on the implementation, DMA access time can be significantly lengthened due to the time required to resolve the actual physical address. If an implementation requires access to a main-memory-resident translation table, the access time can be significantly longer than the time for an untranslated access. Additionally, if each transaction requires multiple memory accesses (e.g., for a table walk), then the memory transaction rate (i.e., overhead) associated with DMA can be high.

To mitigate these impacts, designs often include address translation caches in the entity that performs the address translation. In a CPU, the address translation cache is most commonly referred to as a translation look-aside buffer (TLB). For an I/O TA, the term address translation cache or ATC is used to differentiate it from the translation cache used by the CPU.

While there are some similarities between TLB and ATC, there are important differences. A TLB serves the needs of a CPU that is nominally running one thread at a time. The ATC, however, is generally processing requests from multiple I/O Functions, each of which can be considered a separate thread. This difference makes sizing an ATC difficult depending upon cost models and expected technology reuse across a wide range of system configurations.

The mechanisms described in this specification allow an I/O Device to participate in the translation process and provide an ATC for its own memory accesses. The benefits of having an ATC within a Device include:

- Ability to alleviate TA resource pressure by distributing address translation caching responsibility (reduced probability of “thrashing” within the TA)
- Enable ATC Devices to have less performance dependency on a system’s ATC size
- Potential to ensure optimal access latency by sending pretranslated requests to central complex

This specification will provide the interoperability that allows PCIe Devices to be used in conjunction with a TA, but the TA and its Address Translation and Protection Table (ATPT) are treated as implementation specific and are outside the

scope of this specification. While it may be possible to implement ATS within other PCIe Components, this specification is confined to PCIe Devices and PCIe Root Complex Integrated Endpoints (RCiEPs).

§ Figure 10-1 illustrates an example platform with a TA and ATPT, along with a set of PCIe Devices and RC Integrated Endpoints with integrated ATC. A TA and an ATPT are implementation specific and can be distinct or integrated components within a given system design.

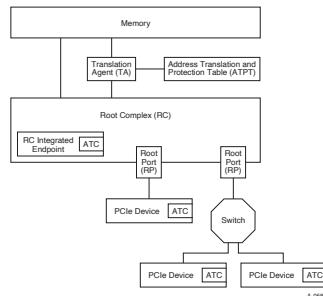


Figure 10-1 Example Illustrating a Platform with TA, ATPT, and ATC Elements §

10.1.1 Address Translation Services (ATS) Overview §

The ATS chapter provides a new set of TLP and associated semantics. ATS uses a request-completion protocol between a Device²⁰⁸ and a Root Complex (RC) to provide translation services. In addition, a new AT field is defined within the Memory Read and Memory Write TLP. The new AT field enables an RC to determine whether a given request has been translated or not via the ATS protocol.

§ Figure 10-2 illustrates the basic flow of an ATS Translation Request operation.

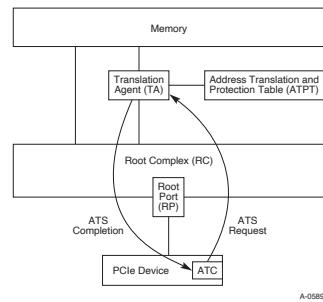


Figure 10-2 Example ATS Translation Request/Completion Exchange §

In this example, a Function-specific work request is received by a single-Function PCIe Device. The Function determines through an implementation specific method that caching a translation within its ATC would be beneficial. There are a number of considerations a Function or software can use in making such a determination; for example:

- Memory address ranges that will be frequently accessed over an extended period of time or whose associated buffer content is subject to a significant update rate

²⁰⁸. All references within this chapter to a Device apply equally to a PCIe Device or a Root Complex Integrated Endpoint. ATS does not delineate between these two types in terms of requirements, semantics, configuration, error handling, etc. From a software perspective, an ATS-capable Root Complex Integrated Endpoint must behave the same as an ATS-capable non-integrated Device.

- Memory address ranges, such as work and completion queue structures, data buffers for low-latency communications, graphics frame buffers, host memory that is used to cache Function-specific content, and so forth

Given the variability in designs and access patterns, there is no single criteria that can be applied.

The Function generates an ATS Translation Request which is sent upstream through the PCIe hierarchy to the RC which then forwards it to the TA. An ATS Translation Request uses the same routing and ordering rules as defined in this specification. Further, multiple ATS Translation Requests can be outstanding at any given time; i.e., one may pipeline multiple requests on one or more TC. Each TC represents a unique ordering domain and defines the domain that must be used by the associated ATS Translation Completion.

Upon receipt of an ATS Translation Request, the TA performs the following basic steps:

1. Validates that the Function has been configured to issue ATS Translation Requests.
2. Determines whether the Function may access the memory indicated by the ATS Translation Request and has the associated access rights.
3. Determines whether a translation can be provided to the Function. If yes, the TA issues a translation to the Function.
 - i. ATS is required to support a variety of page sizes to accommodate a range of ATPT and processor implementations.
 - i. Page sizes are required to be a power of two and naturally aligned.
 - ii. The minimum supported page size is 4096 bytes. ATS capable components are required to support this minimum page size.
 - b. A Function must be informed of the minimum translation or invalidate size it will be required to support to provide the Function an opportunity to optimize its resource utilization. The smallest minimum translation size must be 4096 bytes.
4. The TA communicates the success or failure of the request to the RC which generates an ATS Translation Completion and transmits via a Response TLP through a RP to the Function.
 - a. An RC is required to generate at least one ATS Translation Completion per ATS Translation Request; i.e., there is minimally a 1:1 correspondence independent of the success or failure of the request.
 - i. A successful translation can result in one or two ATS Translation Completion TLPs per request. The Translation Completion indicates the range of translation covered.
 - ii. An RC may pipeline multiple ATS Translation Completions; i.e., an RC may return multiple ATS Translation Completions and these ATS Translation Completions may be in any order relative to ATS Translation Requests.
 - iii. The RC is required to transmit the ATS Translation Completion using the same TC (Traffic Class) as the corresponding ATS Translation Request.
 - b. The requested address may not be valid. The RC is required to issue a Translation Completion indicating that the requested address is not accessible.

When the Function receives the ATS Translation Completion, it either updates its ATC to reflect the translation or notes that a translation does not exist. The Function proceeds with processing its work request and generates subsequent requests using either a translated address or an untranslated address based on the results of the Completion.

- a. Similar to Read Completions, a Function is required to allocate resource space for each completion(s) without causing backpressure on the PCIe Link.
- b. A Function is required to discard Translation Completions that might be “stale”. Stale Translation Completions can occur for a variety of reasons.

As one can surmise, ATS Translation Request and Translation Completion processing is conceptually similar and, in many respects, identical to PCIe Read Request and Read Completion processing. This is intentional to reduce design complexity and to simplify integration of ATS into existing and new PCIe-based solutions. Keeping this in mind, ATS requires the following:

- ATS capable components must interoperate with [PCIe-1.1] compliant components.
- ATS is enabled through a new Capability and associated configuration structure. To enable ATS, software must detect this Capability and enable the Function to issue ATS TLP. If a Function is not enabled, the Function is required not to issue ATS Translation Requests and is required to issue all DMA Read and Write Requests with the TLP AT field set to “untranslated”.
- ATS TLPs are routed using either address-based or ID-based routing.
- ATS TLPs are required to use the same ordering rules as specified in this specification.
- ATS TLPs are required to flow unmodified through [PCIe-1.1] compliant Switches.
- A Function is permitted to intermix translated and untranslated requests.
- ATS transactions are required not to rely upon the address field of a memory request to communicate additional information beyond its current use as defined by the PCI-SIG.

IMPLEMENTATION NOTE: ADDRESS RANGE OVERLAP §

While significant overlap is expected, a system is not required to make Untranslated and Translation/Translated sequences interchangeable for all address ranges. For example, it is permitted for a Root Complex to require Untranslated Requests for interrupt behavior (MSI/MSI-X).

In contrast to the prior example, § Figure 10-3 illustrates an example Multi-Function Device . In this example Device, there are three Functions. Key points to note in § Figure 10-3 are:

- Each ATC is associated with a single Function. Each ATS-capable Function must be able to source and sink at least one of each ATS Translation Request or Translation Completion type.
- If an ATC is associated with a Physical Function implementing a Scalable I/O Virtualization Extended Capability structure, that ATC must either segregate or tag translations associated with each Scalable Device Interface .
- Each ATC is configured and accessed on a per Function basis. A Multi-Function Device is not required to implement ATS on every Function.
- If the ATC implementation shares resources among a set of Functions, then the logical behavior is required to be consistent with fully independent ATC implementations.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

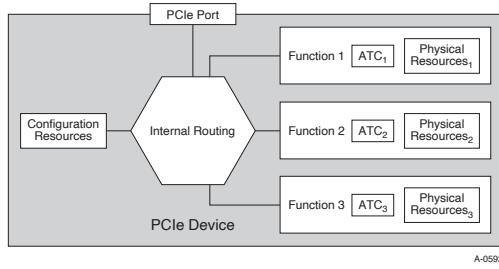


Figure 10-3 Example Multi-Function Device with ATC per Function §

Independent of the number of Functions within a Device, the following are required:

- A Function is required not to issue any TLP with the AT field set unless the address within the TLP was obtained through the ATS Translation Request and Translation Completion protocol.
- Each ATC is required to only be populated using the ATS protocol; i.e., each entry within the ATC must be filled via an ATS Translation Completion in response to the Function issuing an ATS Translation Request for a given address.
- Each ATC cannot be modified except through the ATS protocol. That is:
 - Host system software cannot modify the ATC other than through the protocols defined in this specification except to invalidate one or more translations in an ATC. A Device or Function reset would be an example of an operation performed by software to change the contents of the ATC, but a reset is only allowed to invalidate entries not modify their contents.
 - It must not be possible for host system software to use software executing on the Device to modify the ATC.

When a TA determines that a Function should no longer maintain a translation within its ATC, the TA initiates the ATS invalidation protocol. The invalidation protocol consists of a single Invalidate Request and one or more Invalidate Completions.

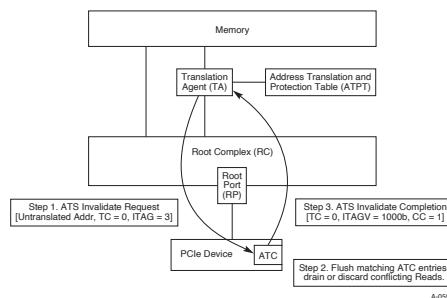


Figure 10-4 Invalidation Protocol with a Single Invalidate Request and Completion §

As § Figure 10-4 illustrates, there are essentially three steps in the ATS invalidation protocol:

1. The system software updates an entry in the tables used by the TA. After the table is changed, the TA determines that a translation should be invalidated in an ATC and initiates an Invalidate Request TLP which is transmitted from the RP to the example Single-Function Device . The Invalidate Request communicates an untranslated address range, the TC, and an RP unique tag which is used to correlate Invalidate Completions with the Invalidate Request.

2. The Function receives the Invalidate Request and invalidates all matching ATC entries. A Function is not required to immediately flush all pending requests upon receipt of an Invalidate Request. If transactions are in a queue waiting to be sent, it is not necessary for the Function to expunge requests from the queue even if those transactions use an address that is being invalidated.
 - a. A Function is required not to indicate the invalidation has completed until all outstanding Read Requests or Translation Requests that reference the associated translated address have been retired or nullified.
 - b. A Function is required to ensure that the Invalidate Completion indication to the RC will arrive at the RC after any previously posted writes that use the “stale” address.
3. When a Function has ascertained that all uses of the translated address are complete, it issues one or more ATS Invalidate Completions.
 - a. An Invalidate Completion is issued for each TC that may have referenced the range invalidated. These completions act as a flush mechanism to ensure the hierarchy is cleansed of any in-flight transactions which may contain references to the translated address.
 - i. The number of Completions required is communicated within each Invalidate Completion. A TA or RC implementation can maintain a counter to ensure that all Invalidate Completions are received before considering the translation to no longer be in use.
 - ii. If more than one Invalidate Completion is sent, the Invalidate Completion sent in each TC must be identical in the fields detailed in § Section 10.3.2 .
 - b. An Invalidate Completion contains the ITAG from Invalidate Request to enable the RC to correlate Invalidate Requests and Completions.

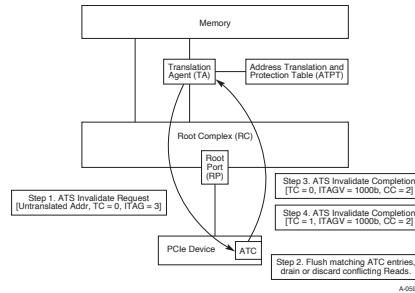


Figure 10-5 Single Invalidate Request with Multiple Invalidate Completions §

10.1.2 Page Request Interface Extension §

ATS improves the behavior of DMA based data movement. An associated Page Request Interface (PRI) provides additional advantages by allowing DMA operations to be initiated without requiring that all the data to be moved into or out of system memory be pinned.²⁰⁹ The overhead associated with pinning memory may be modest, but the negative impact on system performance of removing large portions of memory from the pageable pool can be significant.

PRI is functionally independent of the other aspects of ATS. That is, a device that supports ATS need not support PRI, but PRI is dependent on ATS's capabilities.

Intelligent I/O devices can be constructed to make good use of a more dynamic memory interface. Pinning will always have the best performance characteristics from a device's perspective—all the memory it wants to touch is guaranteed to be present. However, guaranteeing the residence of all the memory a device might touch can be problematic and force a sub-optimal level of device awareness on a host. Allowing a device to operate more independently (to page fault when it requires memory resources that are not present) provides a superior level of coupling between device and host.²¹⁰

²⁰⁹. Locked in place so that it cannot be swapped out by the system's dynamic paging mechanism.

The mechanisms used to take advantage of a Page Request Interface are very device specific. As an example of a model in which such an interface could improve overall system performance, let us examine a high-speed LAN device. Such a device knows its burst rate and need only have as much physical buffer space available for inbound data as it can receive within some quantum. A vector of unpinned virtual memory pages could be made available to the device, that the device then requests as needed to maintain its burst window. This minimizes the required memory footprint of the device and simplifies the interface with the host, both without negatively impacting performance.

The ability to page begs the question of page table status flag management. Typical TAs associate flags (e.g., dirty and access indications) with each untranslated address. Without any additional hints about how to manage pages mapped to a Function, such TAs would need to conservatively assume that when they grant a Function permission to read or write a page, that Function will use the permission. Such writable pages would need to be marked as dirty before their translated addresses are made available to a Function.

This conservative dirty-on-write-permission-grant behavior is generally not a significant issue for Functions that do not support paging, where pages are pinned and the cost of saving a clean page to memory will seldom be paid. However, Functions that support the Page Request Interface could pay a significant penalty if all writable pages are treated as dirty, since such Functions operate without pinning their accessible memory footprints and may issue speculative page requests for performance. The cost of saving clean pages (instead of just discarding them) in such systems can diminish the value of otherwise attractive paging techniques. This can cause significant performance issues and risk functional issues in circumstances where the backing store is unable to be written, such as a CD-ROM.

The No Write (NW flag in Translation Requests indicates that a Function is willing to restrict its usage to only reading the page, independent of the access rights that would otherwise have been granted.

If a device chooses to request only read access by issuing a Translation Request with the NW flag Set and later determines that it needs to write to the page, then the device must issue a new Translation Request.

Upon receiving a Translation Request with the NW flag Clear, TAs are permitted to mark the associated pages dirty. Functions *MUST* not issue such Requests unless they have been given explicit write permission. An example of write permission is where the host issues a command to a Function to load data from a storage device and write that data into memory.

10.1.3 Process Address Space ID (PASID) §

Certain TLPs can optionally be associated with a Process Address Space ID (PASID). This value is conveyed using the PASID TLP Prefix (NFM) or OHC-A (FM). The PASID TLP Prefix is defined in § Section 6.20 .

PASID is permitted for the following types of TLPs:

- Memory Requests (including UIO, AtomicOp, and DMWr Requests) with Address Type (AT) of Untranslated or Translated
- Address Translation Requests (i.e., MRd with AT =01b)
- Page Request Messages
- ATS Invalidate Request Messages
- PRG Response Messages
- Address routed messages in Flit Mode

Usage of PASID for Untranslated Memory Requests is defined in § Section 6.20 . This section describes PASID for the remaining TLPs.

210. The alternative is a private interface between a device and its driver that is used to communicate device state so that the driver can ensure the availability of pinned memory resources.

Each Function has an independent set of PASID values. The PASID field is 20 bits wide, although typically the number of active PASID values will be significantly lower than 2^{20} . PASID values may be allocated sparsely. If the usable width is constrained by the TA or the ATC, the unused upper bits of the PASID value must be 0b. All $2^{\text{usable_width}}$ PASID values are usable. Function hardware is not permitted to assume there are “reserved” PASID values.

An ATC may optionally support Translated Requests with PASID. The Translated Requests with PASID feature is enabled when the Translated Requests with PASID Enable bit is Set. When Translated Requests with PASID Enable is Set, an ATC is permitted to issue Translated Requests with a PASID. When enabled, if the ATC obtained a translation using a Translation Request with PASID, the corresponding Translated Request must carry the same PASID as the Translation Request, the Privileged Mode Requested bit must match the Privileged Mode Access bit in the Translation Response, and the Execute Requested bit is permitted to be Set only if the Execute Permitted bit in the Translation Response was Set. Similarly, if the ATC obtained a translation using a Translation Request without a PASID, the corresponding Translated Request must not carry a PASID. If these rules are not followed the resulting system behavior is undefined.

10.1.4 ATS Memory Attributes §

ATS Memory Attributes (AMAs) provide hints for performing memory operations such as cache management. The AMAs may be supplied to an Endpoint device by a Translation Completion and stored in the ATC. If the ATS Memory Attributes Enable bit is Set, the Endpoint function retrieves the AMAs from the ATC and is permitted to provide AMAs with Memory Read/Write TLPs using the TPH TLP Prefix. This serves as a performance optimization by preventing the TA from having to perform a look-up. The TA and ATC are permitted to support AMAs. If the ATC does not support AMAs but the TA does, then the TA is permitted to perform a lookup to obtain AMAs when processing Memory Read/Write TLPs received from the Endpoint device. A TA that supports AMAs is permitted to always return AMAs in all ATS Translation Completions to all Endpoints.

10.2 ATS Translation Services §

A TA does translations. An ATC can cache those translations. If an ATC is separated from the TA by PCIe, the memory request from an ATC will need to be able to indicate if the address in the transaction is translated or not. The modifications to the memory transactions are described in this section, as are the transactions that are used to communicate translations between a remote ATC and a central TA.

When ~~↓Shadow Functions↓~~ ↑Shadow Functions↑ are enabled, the TA must treat identically all Requests from the “main” Function and its Shadows (See § Section 7.9.25). This usually involves ensuring that system software configures the translation tables used by the TA provide identical answers for the “main” Function and its Shadows. Depending on the architecture of a specific TA, more work may be required (e.g., managing dirty bits, ensuring caches are consistent, etc.). Such work is outside the scope of this specification.

10.2.1 Memory Requests with Address Type §

A Function with an ATC can send Memory Requests that contain either translated or untranslated addresses. The Address Type (AT) field is used to indicate the type of address that is present in the request header (see § Figure 10-6, § Figure 10-7, § Figure 10-8, and § Figure 10-9).

Base 6.4 vs Base 6.3

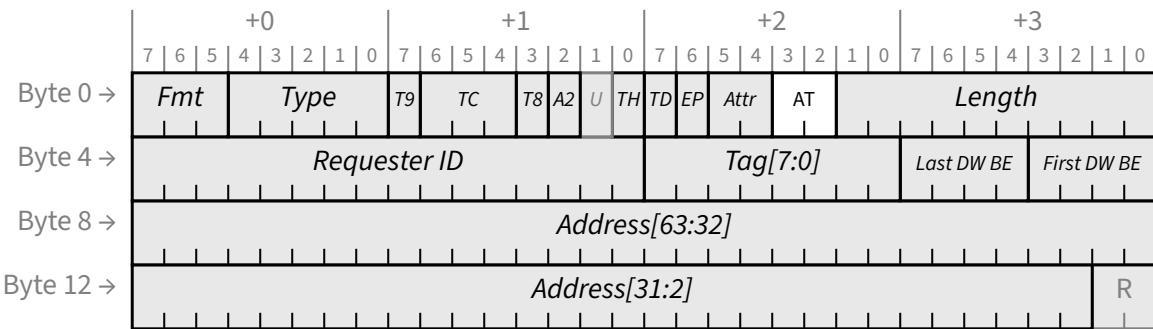


Figure 10-6 Memory Request Header with 64-bit Address Highlighting AT field §

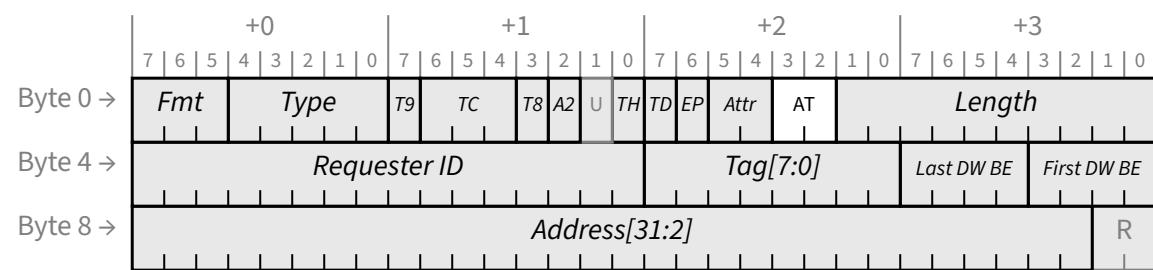


Figure 10-7 Memory Request Header with 32-bit Address Highlighting AT field §

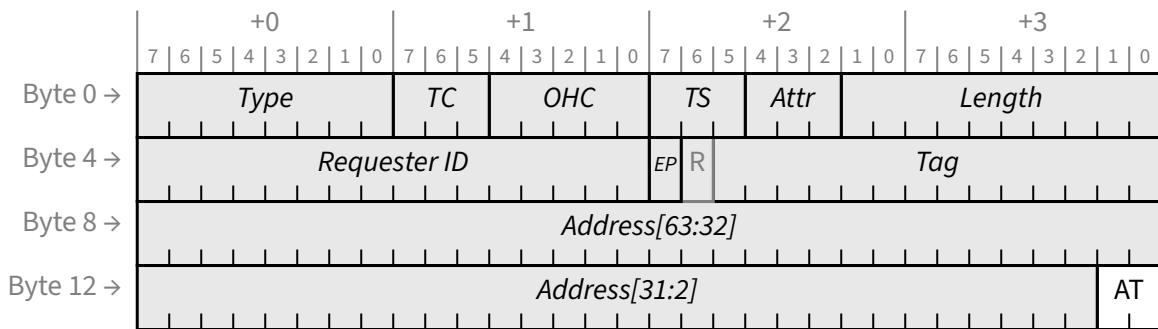


Figure 10-8 Memory Request Header with 64-bit Address Highlighting AT field - Flit Mode §

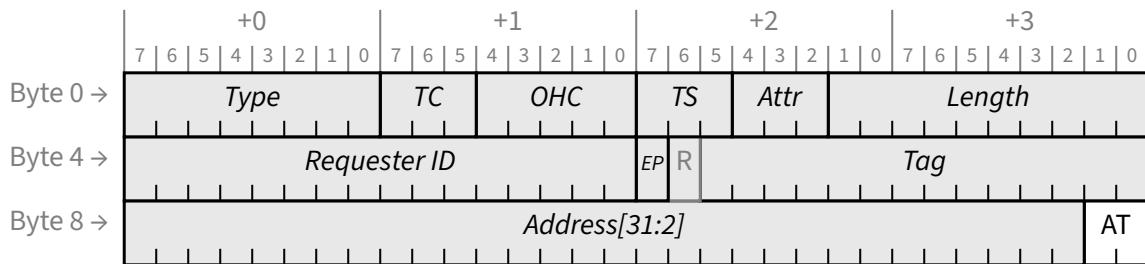


Figure 10-9 Memory Request Header with 32-bit Address Highlighting AT field - Flit Mode §

In NFM, the AT field in the Requests is a redefinition of a reserved field in earlier version of this specification.

Functions that do not implement an ATC will continue to set the AT field to its defined reserved value (00b). Functions that implement an ATC will set the AT field as listed in § Table 10-1.

Table 10-1 Address Type (AT) Field Encodings §

| AT [1:0] Coding | Mnemonic | Meaning |
|-----------------|---------------------|--|
| 00b | Untranslated | A TA may treat the address as either virtual or physical. |
| 01b | Translation Request | The TA will return the translation of the address contained in the address field of the request as a read completion. This value only has meaning for an explicit Translation Request (see § Section 10.2.2). The TA will signal an Unsupported Request (UR) if it receives a TLP with the AT field set to 01b in a Memory Request other than Memory Read. |
| 10b | Translated | The address in the transaction has been translated by an ATC. If the Function associated with the SourceID is allowed to present physical addresses to the system memory, then the TA might not translate this address. If the Function is not allowed to present physical addresses, then the TA may treat this as an UR. |
| 11b | Reserved | The TA will signal an Unsupported Request (UR) if it receives a Memory Request TLP with the AT field set to 11b. |

The AT field is only defined for Memory Requests and in Flit Mode, Address Routed Messages. The field remains reserved for other TLPs.

10.2.2 Translation Requests §

A Translation Request has a format that is similar to that of a Memory Read²¹¹ (MRd TLP Type) in either FM or NFM. The AT field must be set to the value defined for "Translation Request" to differentiate a Translation Request from a normal Memory Read (see § Figure 10-10 , § Figure 10-11 , § Figure 10-12 , and § Figure 10-13). In Flit Mode, OHC-A1 must be included, and the No Write (NW flag is contained in OHC-A1).

Translation Requests have the same completion timeout intervals as Read Requests.

211. Translation Requests are not defined for UIO Memory Reads (UIOMRd TLP Type).

Base 6.4 vs Base 6.3

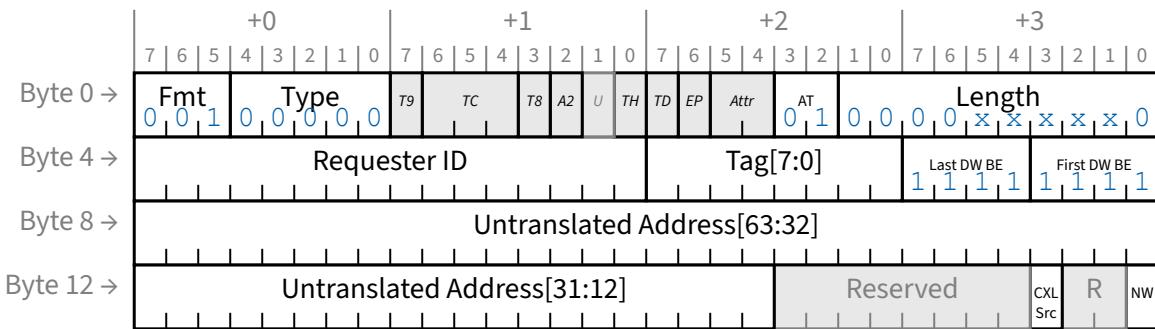


Figure 10-10 Translation Request with 64-bit Address - Non-Flit Mode §

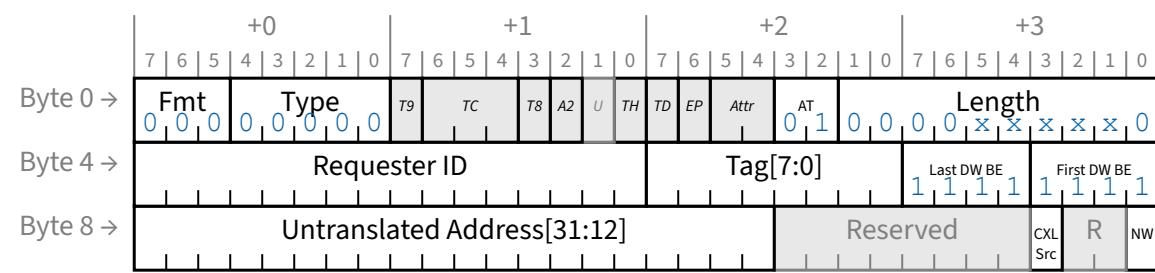


Figure 10-11 Translation Request with 32-bit Address - Non-Flit Mode §

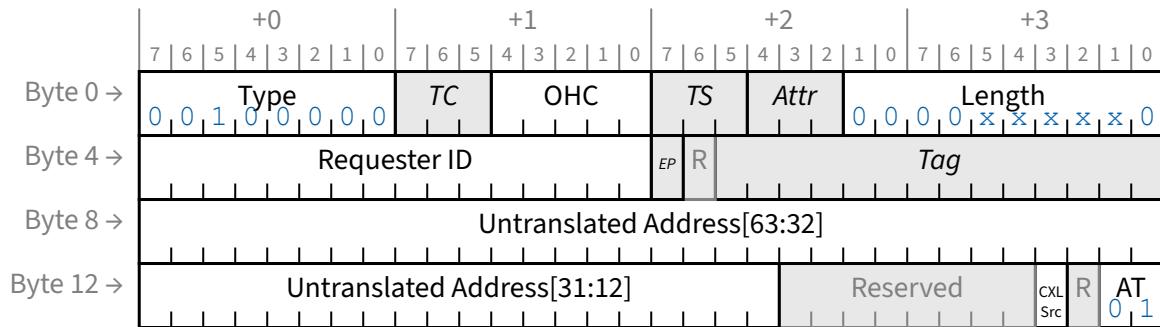
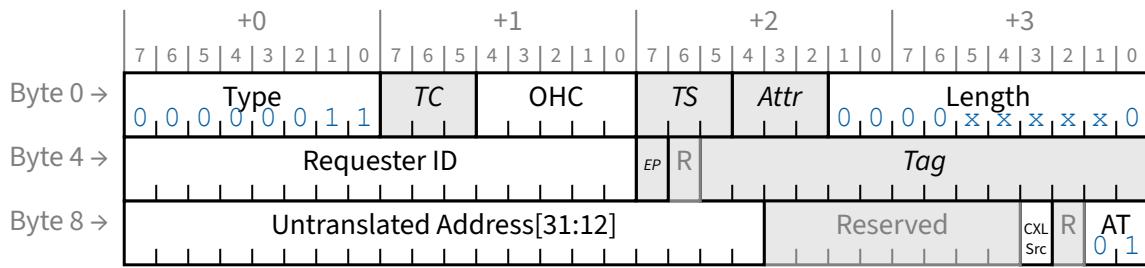


Figure 10-12 Translation Request with 64-bit Address - Flit Mode²¹² §

212. The NW bit is located in OHC-A1.

Figure 10-13 Translation Request with 32-bit Address - Flit Mode²¹³ §

10.2.2.1 Attribute Field §

For a Translation Request, the Relaxed Ordering (RO) bit is applicable and permitted to be Set, where it affects the ordering of its associated Translation Completions. The remainder of the Attr field is Reserved. The Requester of a Translation Request must not depend on the TA to guarantee any specific ordering relationship between Translation Completions and any other Requests or Completions. There are no ordering requirements for a Translation Request. A TA may reorder a Translation Request with respect to any other request.

IMPLEMENTATION NOTE: TRANSLATION REQUEST ORDERING §

Because no ordering can be assumed between Translation Requests and other types of Requests, a Translation Request does not make an effective flushing/ordering primitive.

10.2.2.2 Length Field §

The Length field is set to indicate how many translations may be returned in response to this request. Each translation is 8 bytes in length and represents one or more STUs (Smallest Translation Unit). The maximum setting for the Length field is the RCB of the Root Port as determined by Read Completion Boundary (RCB) in the Link Control Register. The Length field in a Translation Request must always indicate an even number of DWORDs. If Length is set to indicate a value greater than allowed, or if the least-significant bit of the Length field is non-zero, then the resulting handling by the TA is undefined.²¹⁴

If the Length field has a value greater than two, then the Function is requesting translations for a range of memory greater than a single STU. The additional translations, if provided, are assumed to be for sequentially-increasing, equal-sized, STU-aligned regions, starting at the requested address.

10.2.2.3 Tag Field §

The Tag field has the same meaning as in a Memory Read Request.

213. The NW bit is located in OHC-A1.

214. Due to ambiguous text in earlier versions of this document, it should be assumed that an unrecoverable error condition may occur.

10.2.2.4 Untranslated Address Field §

A Translation Request includes either a 32-bit or a 64-bit Untranslated Address field. This field indicates the address to be translated. The TA will make decisions about the validity of the request, based on the address in the translation request. The TA is permitted to return fewer translations than requested, but it will not return more.

When multiple translations are requested, the TA will not return a translation if the range of that translation does not overlap the implied range of the Translation Request (this would only apply to translations after the initial value). The implied range of the Translation Request is $[2^{\text{STU}+12} * (\text{Length}/2)]$ bytes.

The Untranslated Address field in the Translation Request is any address in the range of the first STU. Address bits 11:0 are not present in the Translation Request and are implied to be zero. If a Requester has Page Aligned Request Set (see § Section 7.8.9.2), it must ensure that bits 11:2 are zero. If a Requester has Page Aligned Request Clear, it is permitted to supply any value for bits 11:2.²¹⁵ The TA must ignore bits 11:2 as well as any low-order bits not required to determine the translation.

For example, if using 64-bit addressing for a Function with the Page Aligned Request bit Set that is programmed with an STU of 1 (i.e., 8192-byte pages), bits 63:13 are significant, bit 12 is ignored by the TA and bits 11:0 are implied to be zero.

10.2.2.5 No Write (NW) Flag §

The No Write flag, when Set, indicates that the Function is requesting read-only access for this translation.²¹⁶

The TA may ignore the No Write Flag, however, if the TA responds with a translation marked as read-only then the Function must not issue Memory Write transactions using that translation. In this case, the Function may issue another translation request with the No Write flag Clear, which may result in a new translation completion with or without the W (Write) bit Set.

Upon receiving a Translation Request with the NW flag Clear, TAs are permitted to mark the associated pages dirty. Functions *MUST* not issue such Requests unless they have been given explicit write permission.

In Flit Mode, the NW bit is part of OHC-A1.

10.2.2.6 PASID on Translation Request §

If a Translation Request has a PASID, the Untranslated Address Field is an address within the process address space indicated by the PASID field.

If a Translation Request has a PASID with either the Privileged Mode Requested or Execute Requested bit Set, these may be used in constructing the Translation Completion Data Entry.

The PASID Extended Capability indicates whether a Function supports and is enabled to send and receive TLPs with the PASID.

10.2.2.7 CXL Src §

This bit is assigned for use by [CXL]. In non-CXL systems, this bit is Reserved.

²¹⁵ Note: The Page Aligned Request bit was added in Revision 1.1 of the ATS Specification.

²¹⁶ Note: The No Write Flag was added in Revision 1.1 of the ATS Specification.

10.2.3 Translation Completion §

A Translation Completion (either a Cpl or a CplD) is sent by a TA for each Translation Request. This specification describes the meaning of fields in Translation Completions. Fields not defined in this chapter have the same meanings proscribed for Read Completions in Chapter 2. For a Translation Completion, the Relaxed Ordering (RO) bit is applicable and permitted to be Set if the corresponding Translation Request RO bit was set. The remainder of the Attr field is Reserved.

If the TA was not able to perform the requested translation, a Completion with no data (Cpl) must be returned.

IMPLEMENTATION NOTE: BYTE COUNT FIELD FOR UNSUCCESSFUL TRANSLATION COMPLETIONS WITH NO DATA §

Previous versions of this specification indicated the Byte Count and Lower Address field should be 0000 0000 0000b for Unsuccessful Translation Completions with No Data. It is strongly recommended that implementations do not depend on the Byte Count and Lower Address field being set to any particular value in Unsuccessful Translation Completions with No Data.

~~↑↓Note that in Flit Mode, the Byte Count and Lower Address Fields are not present.↓~~

Errata: Base 6.3
B812△◀

The values and meaning for the Completion Status field are listed in § Table 10-2, where return values other than Success indicate an error.

Table 10-2 Translation Completion Status Codes §

| Value | Status | Meaning |
|------------|--------------------------|--|
| 000b | Success | Indicates that the TA is returning one or more Translation Completion entries (including entries with R=W=0b). |
| 001b | Unsupported Request (UR) | Indicates there is a problem, potentially correctable by system software, that prevents the TA from returning any Translation Completion entries. It is permitted for a Function that receives this Completion code to disable its ATC and not send requests using translated addresses until the ATC is re-enabled. The mechanism a Function receiving this code uses to report this condition is outside the scope of this specification. The TA detecting this error is a "Completer Sending a Completion with UR/CA Status" and shall behave as defined in this specification. |
| 010b | RRS | This value is not allowed in any Completion to a Translation Request initiated by a PCI Express Function. If received by a Function, it is permitted to be handled as a Malformed TLP. |
| 100b | Completer Abort (CA) | The TA was not able to translate the address because of an error in the TA. This status indicates a more serious error condition than UR. Returning CA nominally causes an error to be reported to the device driver associated with the ATC. See AER in this specification. |
| All others | Reserved | A Translation Completion with a Reserved Completion Status value is treated as if the Completion Status was Unsupported Request (001b). |

Completion header fields must be set in accordance with § Section 2.2.9 and § Section 2.3.

Translation Completions must be sent using the same TC as the Translation Request. The Function is not required to verify that the same TC was used. A TA may optionally copy the RO bit of a Translation Request to the Translation

Completion in accordance to the rule specified for the attribute field of completions in § Section 2.2.9 . It is strongly recommended that the TA copy the RO bit. However, if a TA does not copy the RO bit of a Translation Request to the Translation Completion, the TA must clear the RO bit in the Translation Completion.

A Translation Completion with RO Set must follow the ordering rules for Relaxed Ordered Completions as specified in § Section 2.4.1 . A Function that initiates a Translation Request with the RO bit Set but receives the associated Translation Completion with the RO bit Clear is permitted to order associated Translation Completions as if the RO bit is Set.

A Function that initiates a Translation Request with the RO bit Set, must not report an error if the associated Translation Completion has the RO bit Clear.

IMPLEMENTATION NOTE: ATTRIBUTE FIELD COMPATIBILITY IN TRANSLATION COMPLETIONS §

Some implementations of TA may not copy the Attribute field from the request to the completion as required by § Section 2.2.9 , since the Attr field is defined as Reserved in previous versions of this specification. Therefore, the following situations may occur and are handled as follows:

- A TA that does not copy the RO bit (as is typically done for completions as indicated in § Section 2.2.9) by forcing RO to 0 is coupled with a Function conforming to this specification that allows RO to be Set in the request. The Function will accept a Translation Completion with RO Clear and not log an error.
- A TA that conforms to the Attr copy rule (in § Section 2.2.9) is coupled with a Function that does not support RO in Translation Requests. Translation Completions will return with RO Clear as the Function expects.

Therefore, the use of RO is made fully backward compatible. However, it is strongly recommended that the TA support the copy of the RO bit conforming to the rules in § Section 2.2.9 .

The Lower Address field contains a computed value that makes the packet consistent with RCB semantics. If the result is returned in a single packet, Lower Address contains RCB minus Byte Count. If the results are returned in two packets, Lower Address for the first packet contains RCB minus (Total Completion Length * $\frac{\text{Total Completion Length}}{4}$) Lower Address for subsequent packets contains $\frac{\text{Total Completion Length}}{4}$. See § Section 10.2.4 for additional requirements for multiple packet completions.

If the Completion Status field is 000b, then the translation was successful and a data payload will follow the header. The contents of the data payload are shown in § Figure 10-14 .

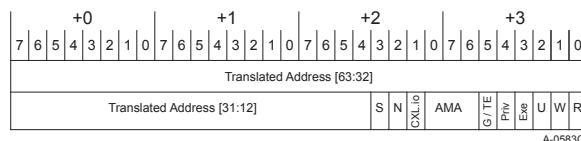


Figure 10-14 Translation Completion Data Entry §

Base 6.4 vs Base 6.3

Table 10-3 Translation Completion Data Fields §

| Field | Meaning | | | | | |
|--|--|--|-----|-----------------------|-------------|---|
| S | Size of translation - This field is 0b if the translation applies to a 4096-byte range of memory. If this field is 1b, then the translation applies to a range of memory that is larger than 4096 bytes (see § Section 10.2.3.1). | | | | | |
| N | Non-snooped accesses - If this field is 1b, then the read and write requests that use this translation must Clear the No Snoop bit in the Attribute field. If it is 0b, then the Function may use other means to determine if <u>No Snoop</u> should be Set. | | | | | |
| CXL.io | This bit is assigned for use by [CXL]. In non-CXL systems, this bit is Reserved. | | | | | |
| AMA | ATS Memory Attributes (AMAs) – If the ATS Memory Attributes Supported bit is Set, the ATC is permitted to cache this AMA value. <table border="1" style="margin-top: 10px;"> <tr> <td style="text-align: center;">AMA</td><td style="text-align: center;">ATS Memory Attributes</td></tr> <tr> <td style="text-align: center;">000b - 111b</td><td>ATS Memory Attribute values (implementation specific). The default value is 000b.</td></tr> </table> | | AMA | ATS Memory Attributes | 000b - 111b | ATS Memory Attribute values (implementation specific). The default value is 000b. |
| AMA | ATS Memory Attributes | | | | | |
| 000b - 111b | ATS Memory Attribute values (implementation specific). The default value is 000b. | | | | | |
| ECN: Base 6.3 XT    | Global Mapping   If this bit is Set, the ATC is permitted to cache this mapping entry in all PASIDs. If Clear, the ATC is permitted to cache this mapping entry only in the PASID associated with the requesting PASID. This bit may only be Set if the associated Translation Request had a PASID.   The presence of this field is mutually exclusive with the TEE Exclusive Memory Attribute (TE) field. TEE Exclusive Memory Attribute   If this bit is Set, the Translated Addresses are associated with TEE memory. If Clear, the Translated Addresses are associated with non-TEE memory (see § Section 10.2.3.10). The presence of this field is mutually exclusive with the Global Mapping (G) field.  If the Global Invalidate Supported is Set in the ATS Capability Register, this field must be treated as the Global Mapping (G) field (see § Section 10.2.3.8).  If the TEE Exclusive Memory Attribute Supported is Set in the ATS Capability Register, this field must be treated as the TEE Exclusive Memory Attribute (TE) field (see § Section 10.2.3.10).  If neither Global Invalidate Supported nor TEE Exclusive Memory Attribute Supported is Set in the ATS Capability Register, this field must be treated as Reserved. | | | | | |
| | Execute Permitted - If this bit is Set, the requesting Function is permitted to execute code contained in the associated memory range. | | | | | |
| | This bit may be Set only if the associated Translation Request had the Execute Requested bit Set. If this bit is Set, R must also be Set. | | | | | |
| | The Priv bit indicates the Privilege level associated with the Exe bit. If Priv is Set, the Exe bit indicates permissions associated with Privileged Mode entities in the Function. If Priv is Clear, the Exe bit indicates permissions associated with Non-Privileged Mode entities in the Function. | | | | | |
| | This value may be cached if R is Set. | | | | | |
| | Privileged Mode Access - If this bit is Set, R, W and Exe refer to permissions associated with Privileged Mode entities. If this bit is Clear, R, W and Exe refer to permissions associated with Non-Privileged Mode entities. | | | | | |

| Field | Meaning |
|-------|--|
| | <p>This bit may only be Set if the associated Translation Request had the <u>Privileged Mode Requested</u> bit Set.</p> <p>This value must be cached any of the R, W or Exe values are cached.</p> |
| U | <p>Untranslated access only - When this field is Set, the indicated range may only be accessed using untranslated addresses, and the Translated Address field of this Translation Completion Data Entry may not be used in a subsequent Read/Write Request with AT set to Translated. This value may be cached if R or W is Set.</p> <p>Future revisions of this specification are expected to deprecate the U bit.</p> |
| R,W | <p>Read, Write - These two fields indicate the transaction types that are allowed for requests using the translation. The encodings are:</p> <ul style="list-style-type: none"> 00b Neither read nor write transactions are allowed. This translation is considered not to be valid. The contents of the Translated Address, N, AMA, $\overline{\text{TE}}$, $\overline{\text{U}}$, and Exe fields are undefined. A translation with this value must not be cached in the ATC (see § Section 10.2.3.5). ECN: Base 6.3 XT$\triangleleft\triangleright$ 01b Write Requests that target this range are allowed, but Read Requests are not unless they are zero-length reads. 10b Read Requests that target this range are allowed (including zero-length reads), but Write Requests are not. 11b Read and Write Requests that target this range are allowed. <p>The Priv bit indicates the Privilege level associated with R and W. If Priv is Set, R and W indicate permissions associated with Privileged Mode entities in the Function. If Priv is Clear, R and W indicate permissions associated with Non-Privileged Mode entities in the Function.</p> <p>If AMA is enabled (AMAE bit is 1b), a zero-length Read Request using a translated address must include AMAs in the TPH TLP Prefix.</p> |

10.2.3.1 Translated Address Field §

If the R and W fields are both Clear, or if U is Set, then the Translated Address field may not be used by the Function for any purpose.

If either the R or W field is Set, and the U field is Clear, then the Translated Address field contains an address that can be used by the Function in a Memory Request with the AT field set to Translated and the Function may cache the Translated Address. When cached, the R and W fields must be stored with the same value as the Translation Completion entry. The address that is cached must be a subset of the address range indicated in the Translation Completion (the subset may include the entire range).

While the Translated Address is cached in the Function's ATC, it shall not be possible for the Function to modify the entry other than to delete it. The entry must be deleted from the ATC when an Invalidate Request is received that has an indicated range that overlaps any portion of the cached address.

A Function is not allowed to make an entry into its ATC unless the entry is in a Translation Completion and the E (Enable) field within the ATS Capability is Set. Entries in an ATC cache that are written before the E field is Set must not be used in Memory Request. They must either be invalidated when the E field is Set or ignored and not used.

10.2.3.2 Translation Range Size (S) Field §

If S is Set, then the translation applies to a range that is larger than 4096 bytes. If S = 1b, then bit 12 of the Translated Address is used to indicate whether or not the range is larger than 8192 bytes. If bit 12 is 0b, then the range size is 8192 bytes, but it is larger than 8192 bytes if Set. If S = 1b and bit 12 = 1b, then bit 13 is used to determine if the range is larger than 16384 bytes or not. If bit 13 is 0b, then the range size is 16384 bytes, but it is larger than 16384 bytes if Set.

Low-order address bits are consumed in sequence to indicate the size of the range associated with the translation.

Note: This encoding method is also used to indicate the size of the memory range being invalidated.

Examples for different translation sizes are shown in § Table 10-5 .

Table 10-5 Examples of Translation Size Using S Field §

| Address Bits | | | | | | | | | | | | | | | | | | | S | Translation Range Size in Bytes | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------------------------|-----|------|
| 63:32 * | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | | |
| x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 | 4 K | |
| x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 | 1 | 8 K |
| x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 | 1 | 16 K |
| x | x | x | x | x | x | x | x | x | x | x | x | x | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 M | |
| x | x | x | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 G | |
| x | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 G | |

Note:

* Upper address bits are used to indicate the size for ranges larger than 4 GB.

The size field is set to indicate the range size in multiples of 4096 bytes regardless of the setting of STU. For example, if STU is set to indicate that the minimum translation is 8192 bytes, then S should be Set on all translation returned in a Translation Completion and in all Invalidate Requests. If STU is set to indicate a 16384-byte minimum, then S and bit 12 would both be Set in all translation and invalidate ranges.

If S is Set and bits 63:12 are all 1b, then the behavior is undefined. If S is Set and bit 63 is 0b, and bits 62:12 are all 1b, then the request is to invalidate all translations.

If a Function receives a Translation Completion with a Translation Size field smaller than the Function's programmed STU value, it shall treat the Translation Completion as if it had Completion Status UR.

10.2.3.3 Non-snooped (N) Field §

This field is Set to indicate that Read and Write Requests that target memory in the range of this translation must Clear the No Snoop Attribute bit in the Request header. When this field is 0b, the Function is allowed to Set the No Snoop Attribute bit in a Function-specific manner.

Note: When this field is Cleared, the Function is not allowed to Set No Snoop in a Memory Request if the Enable No Snoop field in the Device Control register is Cleared.

The N bit may be cached by the ATC if either R or W is Set.

When U is Set, the meaning of this field is undefined, and the TA may set this field to any value. A translation has a single value for the N field that is not affected by privilege level. An ATC is permitted to cache the N field without regard to the value of the Priv bit.

10.2.3.4 Untranslated Access Only (U) Field §

Future revisions of this specification are expected to deprecate the U bit.

This field is Set when the Function is not allowed to access the implied range of memory using a translated address (the range is implied by the untranslated address in the Translation Request and the offset of the translation in the Translation Completion). The Function may use untranslated addresses to access the range as long as the accesses are allowed by the R and W fields. The Function may cache this translation value if either R or W is Set. If the U field is Set, the Translated Address field in the translation is not necessarily a valid memory address and the Function may not use the value in a Read or Write Request with AT set to Translated.

Note: One of the possible uses of this field is to avoid unnecessary invalidations. If a Function uses translated requests for some portions of memory, but not others, then the U field can be used on the portions for which translated requests are not used. When a translation changes if the U field is Set, then it will not necessarily be required that an Invalidate Request be sent to the Function. An example of this use is a Function with a ring buffer that is used for commands. The ring buffer may be allocated for a long period of time and have very high \downarrow re-use \downarrow \uparrow reuse \uparrow (locality). For this reason, it is useful for the Function to use translated addresses in its memory request that target the command buffer. The same Function might access data buffers that have poor locality and low reuse. Accesses to the data buffers might best be handled by using untranslated Requests. Setting the U field for the data buffer translations ensures that the Function will not attempt to use a translated value to access the data buffer so, when the data buffer mappings are changed, no Invalidate Request is required. When U is Set and either R or W is Set, the ATC is permitted to cache U, R, W Exe, and Priv, as well as the Translation Range Size (see § Section 10.2.3.2). An Invalidate Request is required if these values change.

10.2.3.5 Read (R) and Write (W) Fields §

These fields indicate if the returned translation value may be used in a read or write memory request. The ATC is not permitted to issue a non-zero length read request using the translation value if the R field is Cleared. The ATC is not permitted to issue a write request using the translation value if the W field is Cleared. The ATC may not issue any type of request using the translation value if neither the R nor W fields are Set. If both R and W fields are Cleared, the range of the translation is still indicated, but the meaning of the other values in the translation is undefined.

Note: The range of a Translation entry is indicated even if R = W = 0b in order to allow a “hole” in the Translation Completion. For example, if the Translation Request has a Length of six DWs, then up to three translations could be included in the Translation Completion. The first and third translations may have Set R or W but the second could have R = W = 0b. To avoid ambiguity about the size of the indicated gap, the range of the gap is indicated in the Translation Completion even if R = W = 0b.

The R = 0b, W = 0b state is used to indicate that the ATC is not permitted to use address field in the translation to form a translated address value for a subsequent request.

When the host changes a translation in the TA from not valid to valid, the host is not required to send an invalidation indication to the ATC. As such the ATC will not be notified of such changes and thus the ATC is not permitted to cache a translation value of $R = W = 0b$.

When the host changes permissions associated with a translation in the TA, to grant additional permission (e.g., $R = 1, W = 0b$ to $R = W = 1b$), the host is not required to send an invalidation indication to the ATC. As such, the ATC will not be notified of such changes. When the ATC requires permissions greater than the cached ATC entry, the ATC is permitted to request a fresh translation.

When the host changes permissions associated with a translation in the TA, to remove permission (e.g., R = W = 1b to R = 1b, W = 0b), the host is required to send an invalidation indication to the ATC. The subsequent Invalidate Completion tells the host that the ATC has stopped using the previously granted permissions.

If no table entry is found for the requested address, the TA must return a CplD with a single translation value with R = W = 0b.

Note: Implementations should not assume that receiving a translation response with the R or W bits Set (*independent of the value of the U bit*) implies that a subsequent read or write request with the same **untranslated** address will succeed. Although it may be possible for a device and its controlling software to ensure this property, the method for doing so is outside the scope of this specification.

When ACS Direct Translated P2P is enabled, Translated and Untranslated requests may take different paths between Requester and Completer. As such, PCI Express Ordering between Translated and Untranslated requests is not guaranteed. For transaction sequences that require ordering, Functions should avoid using a mixture of Translated and Untranslated Requests.

The Priv bit is used to qualify R and W. If Priv is Set, R and W indicate permissions granted to Privileged Mode entities in the Function. If Priv is Clear, R and W indicate permissions granted to Non-Privileged Mode entities in the Function. The R and W values for the two privilege levels are independent. The ATC must not assume any correlation between the Privileged Mode and Non-Privileged Mode permissions associated with a translation.

10.2.3.6 Execute Permitted (Exe) §

If Exe is Set, the requesting Function is permitted to execute code in the implied range of memory. If Exe is Clear, the requesting Function is not permitted to execute code in the implied range of memory.

The definition of what it means for a Function to execute code is outside the scope of this specification. Various system components may have different instruction sets. Behavior within the requesting Function when it attempts to execute code that is not permitted by this bit is outside the scope of this specification.

The Exe bit may only be Set if the TA supports Execute permissions, the associated Translation Request had an effective value of 1b for the Execute Requested bit²¹⁷ and R is Set in the Translation Completion Data Entry. Otherwise, the Exe bit must be Clear.

This value may be cached if R is Set.

The Priv bit is used to qualify the Exe bit. If Priv is Set, the Exe bit indicates permission granted to Privileged Mode entities in the Function. If Priv is Clear, the Exe bit indicates permission granted to Non-Privileged Mode entities in the Function. The Exe bit values for the two privilege levels are independent. The ATC must not assume any correlation between the Privileged Mode and Non-Privileged Mode permissions associated with a translation.

Functions may optionally check that:

- If the Execute Requested bit is Clear in a Translation Request, the Exe bits in the associated Translation Completion Data Entries are also Clear.
- If Exe is Set, R is also Set.

If either optional check fails, the Function shall signal Unexpected Completion (UC). These checks are independently optional.

²¹⁷. The effective value of the bit is 0b unless the bit in the PASID TLP Prefix (NFM) or OHC-A1 is 1b and usage of the bit is enabled for the request.

10.2.3.7 Privileged Mode Access (Priv) §

If Priv is Set, R, W, and Exe refer to permissions granted to entities operating in Privileged Mode in the requesting Function. If Priv is Clear, R, W, and Exe refer to permissions granted to entities operating in Non-Privileged Mode in the requesting Function.

The meaning of Privileged Mode and Non-Privileged Mode and what it means for an entity to be operating in Privileged Mode or in Non-Privileged Mode depends on the protection model of the system and is outside the scope of this specification.

Behavior is outside the scope of this specification when an entity in the requesting Function attempts to access memory that it is not permitted to access.

The Priv bit may only be Set if the TA supports Privileged Mode and the associated Translation Request had a PASID TLP Prefix (NFM) or OHC-A1 with an effective value of 1b for the Privileged Mode Requested²¹⁸ bit. Otherwise, the Priv bit must be Clear.

The Privileged and Non-Privileged Mode versions of R, W and Exe are independent. An ATC may cache either or both versions of R, W and Exe. An ATC that receives a translation with R=W=0b for one privilege level may not assume anything about what it might receive for the other privilege level.

This value may be cached if R or W is Set. This value must be cached when the corresponding R, W, or Exe values are cached.

Note: Since the Priv bit is Set only when the requesting Function Sets the Privileged Mode Requested bit, Functions that never set that bit should always receive the Priv bit Clear and thus don't need to cache it.

Functions may optionally check that when the Privileged Mode Requested bit is Clear in a Translation Request, the Priv bits in the associated Translation Completion Data Entries are also Clear. If this optional check fails, the Function shall signal Unexpected Completion (UC).

IMPLEMENTATION NOTE: EXECUTE PERMISSION AND PRIVILEGE MODE ENFORCEMENT §

The requesting Function determines whether a particular Memory Request needs Execute permission or is associated with a Privileged Mode or Non-Privileged Mode entity. The ATC implements the protection checks indicated by the Exe and Priv bits.

10.2.3.8 Global Mapping (Global) §

If Global is Set, the requesting Function is permitted to create a Global Mapping entry in the ATC for this translation. If Global is Clear, the requesting Function is not permitted to create a Global Mapping entry in the ATC for this translation. Global Mapping entries apply to all PASIDs of the Function. They permit the ATC to reduce the number of translation requests needed and to reduce the memory needed for caching the results.

A Function is permitted to ignore this bit and always create non-Global Mapping entries in the ATC. This could result in multiple translations being requested for the same Untranslated Address under different PASIDs.

²¹⁸. The effective value of the bit is 0b unless the bit in the in the PASID TLP Prefix (NFM) or OHC-A1 is 1b and usage of the bit is enabled for the request.

Functions that use this bit must also have the ~~↑↓Global Invalidate Supported↓~~ ↑↓Global Invalidate Supported↑ bit Set (see § [Section 10.5.1.2](#)).

IMPLEMENTATION NOTE: TLP LENGTH, ADDRESS, AND BYTE OFFSET VALUES FOR TRANSLATION REQUEST COMPLETIONS §

The intention behind the rules for Translation Completions containing multiple translations is to make the TLPs look similar to those contained in a Memory Read Completion.

For single TLP Translation Completions, the goal is to make the Completion look as though it is a Memory Read Completion that ends on an RCB. As such, the Length and Byte Count will indicate the same value, and the Lower Address will contain the value that makes the TLP appear to end on an RCB.

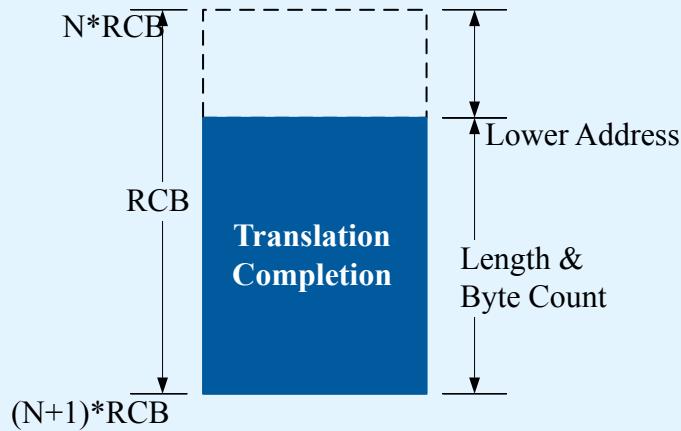


Figure 10-15 Example Translation Completion with 1 TLP §

To intermediate components (which do not track the transaction) this TLP will be indistinguishable from a Memory Read Completion ending on an RCB.

For Translation Completions consisting of two TLPs, the goal is to make the Completion look as though it is a Memory Read Completion that crosses an RCB. As Such, the first Completion TLP will contain Lower Address & Length values which make the TLP appear to end on an RCB. The Byte Count of the first TLP will indicate the total length of all the Translation Completions sent in this transaction. For the second TLP, the Length and Byte Count fields will indicate the same value, and the Lower Address value will be 0.

Base 6.4 vs Base 6.3

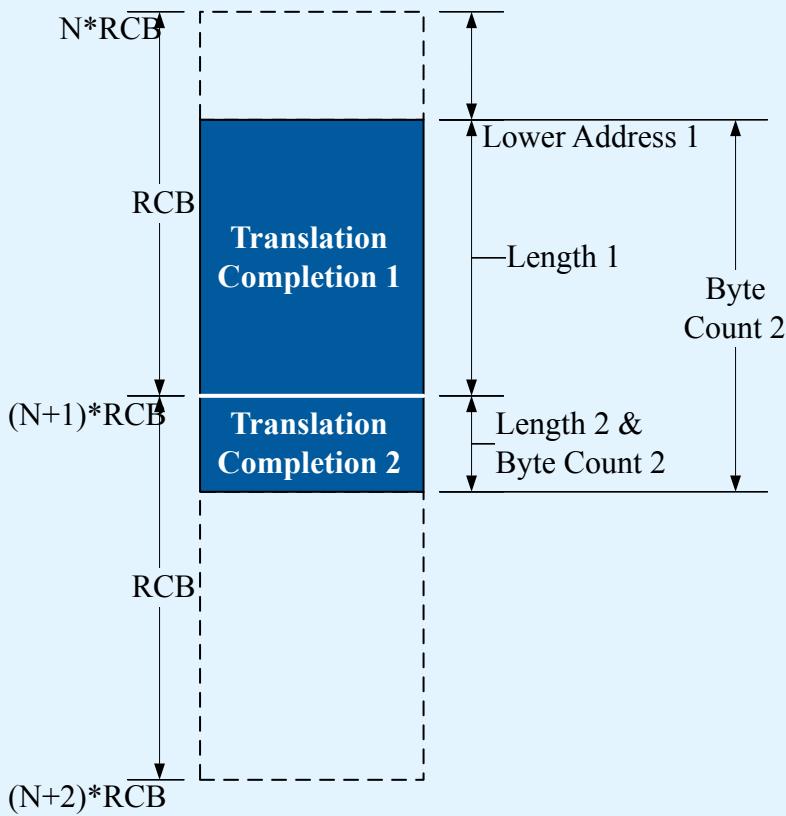


Figure 10-16 Example Translation Completion with 2 TLPs §

To intermediate components (which do not track the transaction) this Completion transaction will be indistinguishable from a Memory Read Completion that crosses an RCB.

Note that the Length field is measured DWORDS, whereas the Lower Address and Byte Offset fields are measured in Bytes.

Provided the TLPs are properly formatted, A TA may choose to split the Translation Completion between any 2 Translation Completion Data Entries. Because an ATC cannot request more translations than can fit within a single RCB, the architectural maximum number of Translation Completion Data Entries can be sent in a single Completion TLP.

10.2.3.9 ATS Memory Attributes §

AMA values are associated with the Completer and are implementation specific. AMA values are opaque to the Requester, the Requester's ATC and the TA.

The method for detecting support of, and for enabling, the AMA mechanism in a TA is implementation specific. When the AMA mechanism is not enabled in both the Requester and the TA, the default AMA value is 000b. Behavior of the Completer for this case is implementation specific.

An ATC that supports AMA, caches AMA values along with other ATC information (Translated Address, Read, Write permission etc.) If the AMA value changes, for a page that could be cached in an ATC, software must issue an ATS Invalidate Request covering that page to ensure the ATC flushes the stale AMA value.

IMPLEMENTATION NOTE: ATS MEMORY ATTRIBUTES §

ATS Memory Attributes (AMAs) are used with bus transactions on the Internal Coherent Bus (ICB) of devices as shown in the figure below. AMAs simplify processing at the TA. AMAs are simply passed through to the ICB avoiding a look-up at the TA.

AMAs may be supplied by a Translation Completion and stored in the ATC. The Endpoint may provide AMAs in Memory Read/Write TLPs using a TPH TLP Prefix. AMAs may also be provided using Vendor-Defined Messages, however such methods are outside the scope of this specification.

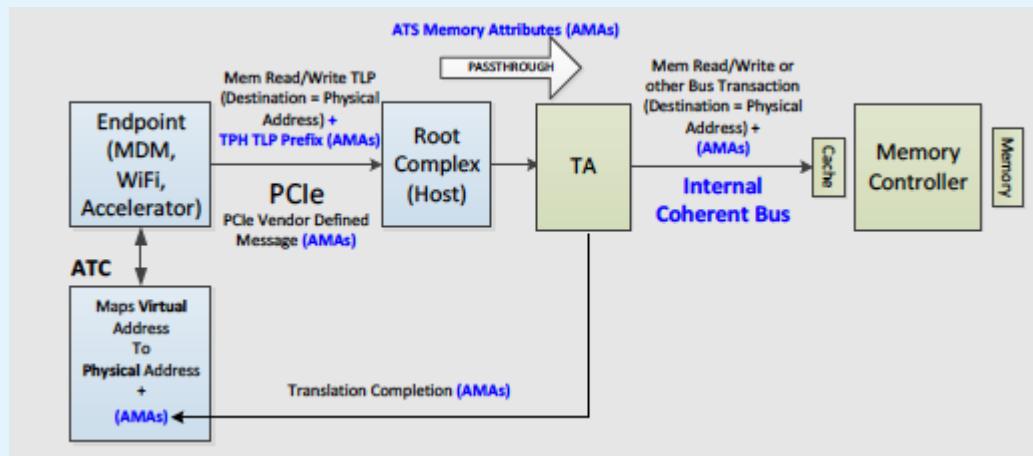


Figure 10-17 ATS Memory Attributes Example §

AMAs provide hints on performing cache management as well as behavior on the ICB for a particular transaction. For example, AMAs may hint:

- Data is cacheable or non-cacheable.
- Write-hit policy. “Write back” may indicate that on a cache hit, update the cache, do not update memory. Mark the cache as dirty.
- Write-miss policy. “Write allocate” may indicate that on a cache miss, update memory, bring the data block to the cache (allocate).

AMAs are similar to TLP Processing Hints but should be considered independent. Hence all permutations of AMAs and TLP Processing Hints are permitted. However, AMAs may convey different semantics than what is provided by TLP Processing Hints. The cache behavior conveyed by AMAs and TLP Processing Hints may not align.

Additionally, AMAs may convey semantics not addressed by TLP Processing Hints. For example, handling of transactions on the ICB when caching is not in use, such as, “reordering permitted/not permitted”, “early write acknowledgement permitted/not permitted”.

AMAs may be transmitted in peer-to-peer transactions. It is recommended that an application programming interface (API) be provided to enable AMAs to be configured into the Translation Agent (TA). Note: the TA might not be collocated with the Completer.

An example of memory attributes such as “strongly ordered device normal” may be found at <http://infocenter.arm.com>.

10.2.3.10 ↑↑TEE Exclusive Memory Attribute (TE)↑ §

ECN: Base 6.3

XTΔ↔

↑↑Trusted Execution Environments (TEEs) are associated with two types of memory: TEE memory and non-TEE memory. TEE memory includes the memory used for the TEE's code, data, and execution state. Additionally, the memory in TEE Device Interfaces (TDIs) that are associated with the TEE and have the IS_NON_TEE_MEM attribute Cleared also falls under the TEE memory. All other memory related to the TEE is classified as non-TEE memory. TEE memory is protected for confidentiality and integrity by restricting access only to entities within the TEE's Trusted Computing Base (TCB). Conversely, non-TEE memory is accessible to entities outside the TEE's TCB, and its contents are not considered trustworthy by the TEE.↑

↑↑The TEE Exclusive Memory Attribute (TE) bit, when Set indicates that the Translated Addresses correspond to locations in TEE memory. When the TEE Exclusive Memory Attribute (TE) bit is Cleared, it signifies that the Translated Addresses correspond to locations in non-TEE memory.↑

↑↑The rules for the use of the TEE Exclusive Memory Attribute (TE) bit by TDIs are specified in § Section 11.4.10.↑

↑↑Functions that support use of this bit must have the TEE Exclusive Memory Attribute Supported bit Set in the ATS Capability Register (see § Section 10.5.1.2).↑

10.2.4 Completions with Multiple Translations §

An ATC is allowed to request that the TA provide translations for a virtually contiguous range of addresses. It does this by setting the Length field in the Translation Request to a value that is two times the number of requested translations as long as the request size (Total Completion Length * 4) is not larger than Read Completion Boundary (RCB) in the Link Control Register.

If multiple translations are requested, the TA may return one or more translations as long as the number of translations does not exceed the number of requested translations. It is not an error for the TA to return fewer translations than requested and no error indication is sent unless there is an error in accessing the data.

If the Translation Completion contains multiple translations, all translations must have the same indicated size. Also, successive translations must apply to the virtual address range that abuts the previous translation in the same completion.

If a translation has both R = 0b and W = 0b, the TA must still set the Size field and the lower bits of the Translated Address field used to encode the completion size to appropriate values.

Each translation in a Translation Completion will have some overlap with the implied memory range of the Translation Request (see § Section 10.2.2).

A successful Translation Completion must consist of one or two CplDs. Each CplD must contain an integral number of Translations (i.e., Length must be a multiple of 2).

The TA is permitted to choose:

- the number of translations it returns for each Translation Request (e.g., Byte Count of the first or only CplD)
- if it returns more than one translation, whether it uses one or two CplDs
- if it returns two CplDs, how many translations are returned in each CplD

The Byte Count and Length fields in each CplD is used to convey these choices to the ATC. The Lower Address field should not be needed by the ATC (its value is computed as defined in § Section 10.2.3 to satisfy RCB rules but the field otherwise conveys no additional information).

- If a Translation Completion CplD has a Byte Count that is greater than four times the Length field, then this is the first of two CplDs for the transaction.
- If a Translation Completion CplD has a Byte Count that is equal to four times the Length field, then this is the second or only CplD for the request.

Note: There are multiple reasons that the TA may truncate the results of the completion. For example, the request might ask for a range of addresses, not all of which are defined. This could occur if the first translation is valid but located at the end of a page of translations. The TA, in looking up the next page of translations, may find that the page is not valid so the addresses are not valid. The range of addresses that are valid would be returned and no error indicated. When truncating a Translation Completion the TA is not allowed to pad the response with invalid entries (R = 0b, W = 0b).

Note: There are multiple reasons that the TA may break a Translation Completion into multiple TLPs. As an example, if the virtual address of the Translation Completion resolves to a table access that crosses an implementation specific address boundary, the completion to the TA may be broken into two completions. Rather than require that the TA accumulate the results, the TA is permitted to send each portion of the Translation Completion to a Function when it is received from memory.

10.3 ATS Invalidation §

ATS uses the messages shown in this section to maintain consistency between the TA and the ATC. This specification assumes there is a single TA associated with each ATC. The TA (in conjunction with its associated software) must ensure that the address translations cached in the ATC are not stale by issuing Invalidate Requests.

10.3.1 Invalidate Request §

When a translation is changed in the TA and that translation might be contained within an ATC in a Function, the TA (in conjunction with its associated software) must send an Invalidate Request to the ATC to maintain proper synchronization between the ATPT and the ATC. An Invalidate Request is used to clear a specific subset of the address range from the ATC. Invalidate Requests are constrained to cover power of 2 multiple of 4096-byte pages.

The format of an Invalidate Request is shown in § Figure 10-18 and § Figure 10-19 .

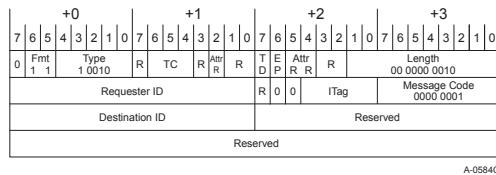


Figure 10-18 Invalidate Request Message - Non-Flit Mode §

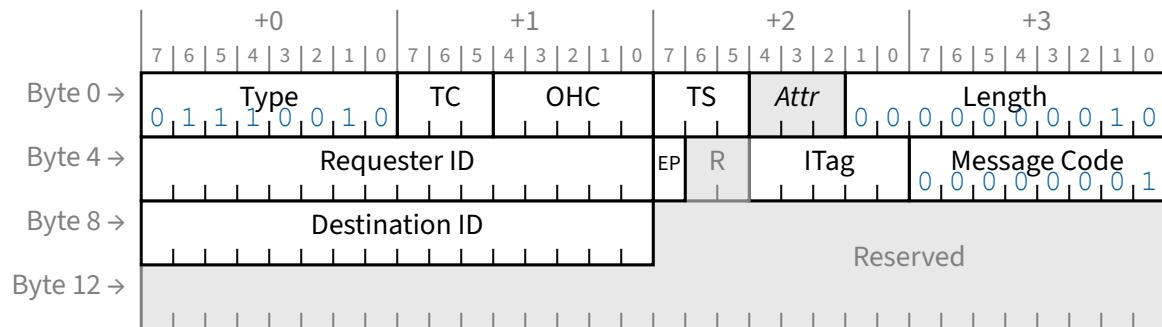


Figure 10-19 Invalidate Request Message - Flit Mode §

The Invalidate Request is a MsgD transaction with 64 bits of data. Invalidate Request messages may be sent in any TC. Invalidate Request Messages optionally include a PASID (see § Section 10.3.8)

- In Non-Flit Mode, PASID is included when a PASID TLP Prefix is present. The Execute Requested and Privileged Mode Requested bits in this PASID TLP Prefix are Reserved.
- In Flit Mode, PASID is included when OHC-A4 is present and the PV is Set. PASID is not included if either OHC-A4 is not present or if PV is Clear. OHC-A4 also contains Destination Segment if DSV is Set²¹⁹.

The ITag field is used by the TA to uniquely identify Invalidate Requests it issues. A TA must ensure that once an ITag is used, it is not reused for the same Destination ID until either released by the corresponding Invalidate Completions or by a vendor-specific timeout mechanism (see below). A TA is permitted to use the same ITag value for Invalidate Requests that target different Destination IDs. See the Invalidate Queue Depth field in the ATS Extended Capability for additional considerations.

The address range specified in an Invalidate Request may span one or more STU 4096-byte pages. Invalidation ranges are required to be naturally aligned and should not be smaller than STU 4096-byte pages. Upon receiving an Invalidate Request with a range less than STU an ATC may either (1) signal an Unsupported Request (not recommended) or (2) round the range of the request up to a value greater than or equal to the STU.

IMPLEMENTATION NOTE: INVALIDATE COMPLETION TIMEOUT §

Devices should respond to Invalidate Requests within 1 minute (+50% -0%). Having a bounded time permits an ATPT to implement Invalidate Completion Timeouts and reuse the associated ITag values. ATPT designs are implementation specific. As such, Invalidate Completion Timeouts and their associated error handling are outside the scope of this specification.

The content of the payload is the untranslated address range to be invalidated. The payload format is shown in § Figure 10-20 .

²¹⁹. Invalidate Request Messages are issued by the TA through a specific RP. Since everything below a given RP is part of the same segment, the Destination Segment field in the message is not required unless needed by the RC to determine the appropriate RP.

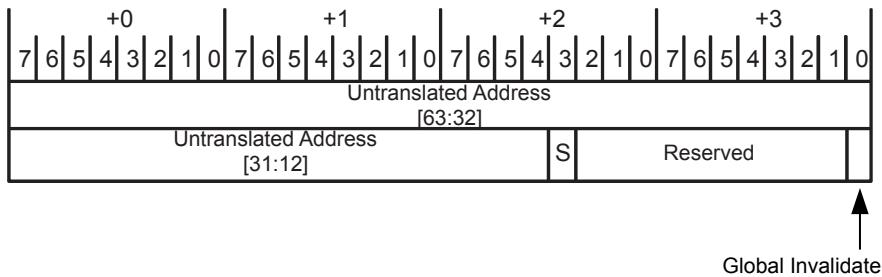


Figure 10-20 Invalidate Request Message Body §

The S field is used to indicate if the range being invalidated is greater than 4096 bytes. Its meaning is the same as for the Translation Completion (see § Section 10.2.3.1 and § Section 10.2.3.2).

The Global Invalidate bit indicates that the Invalidate Request Message affects all PASID values (see § Section 10.3.8). This bit is Reserved unless the Invalidate Request has a PASID. The bit is ignored by the ATC if Global Invalidate Supported bit is Clear (see § Section 10.3.8).

IMPLEMENTATION NOTE: INVALIDATE REQUESTS AND FUNCTION LEVEL RESET & DEVICE POWER STATE TRANSITIONS §

Invalidate Requests received while a Function is undergoing Function Level Reset, or is in (or transitioning to) non-D0 device state, may be dropped by the Function. Similarly, Invalidate Requests already received but pending at the time of receiving initiate FLR or D-state transition request may be dropped by the Function.

System software can avoid ATS invalidation race conditions on Function Level Reset and Device Power State transitions in a variety of ways. For example:

1. When disabling ATS on a Function, system software quiesces ATS invalidations to the Function (i.e., either responses are received for all Invalidate Requests issued to the Function, or any pending Invalidate Requests to the Function have timed out).
2. Software ensures no Invalidate Requests are issued to a Function when its ATS capability is disabled.
3. Before initiating the FLR (or Device power state transitions), software disables ATS as described in item 1. above.

10.3.2 Invalidate Completion §

When a Function completes an invalidate operation, it will send one or more Invalidate Completion messages to the TA. These messages must be tagged with information extracted from the Invalidate Request to enable the TA to associate the Invalidate Completions with the Invalidate Request.

The format of the Invalidate Completion message is shown in § Figure 10-21 and § Figure 10-22.

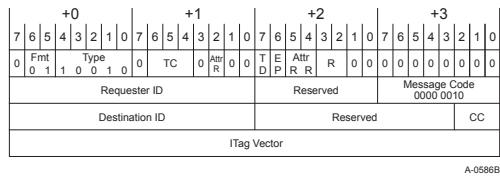


Figure 10-21 Invalidate Completion Message Format - Non-Flit Mode

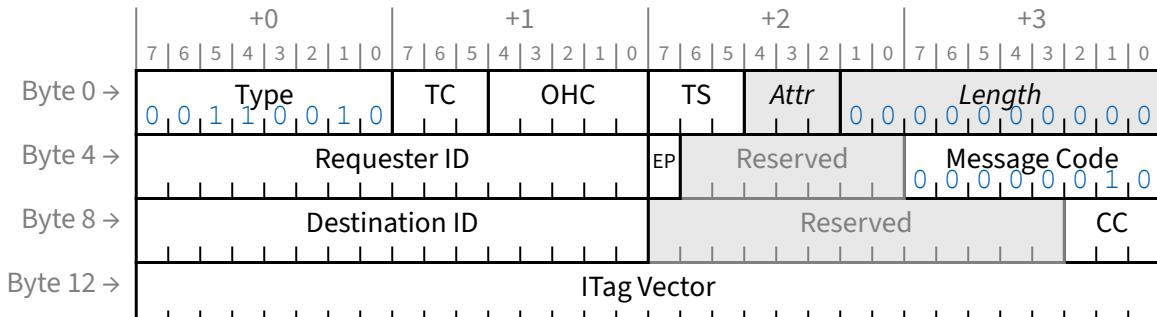


Figure 10-22 Invalidate Completion Message - Flit Mode

The Invalidate Completion message is a Msg transaction routed by ID. The Requester ID field of the Invalidate Completion message is set to the Requester ID of the Function containing the ATC. The **↓↑Device↓** **↑↑Destination↑** ID field of the Invalidate Completion is set to the Requester ID of the TA. The ATC may derive the Requester ID of the TA from the Requester ID field of the corresponding Invalidate Request. Alternatively, since the ATC is only associated with a single TA, the ATC may sample and store the Requester ID from the first Invalidate Request following a Fundamental Reset or FLR. Subsequent Invalidate Completion messages may use this value to set the **↓↑Device↓** **↑↑Destination↑** ID field of Invalidate Completion messages.

Errata: Base 6.3
B858△◀

In Flit Mode, Invalidate Completion messages optionally contain OHC-A4 when a Destination Segment is needed (e.g., the TA and the ATC are in different segments). The PASID is Reserved. PV must be 0b.

The Completion Count (CC) field indicates the number of individual Invalidate Completion messages that must be sent for the associated Invalidate Request. Setting the CC field to 0 indicates that eight responses must be sent. The TA is responsible for collecting all the responses associated with a given Tag before considering the corresponding Invalidate Request to be complete.

Invalidate Completion messages may be sent on any TC, independent of the TC the originating Invalidate Request was received. This enables implementations to utilize the Invalidate Completion to push outstanding transactions to the TA to guarantee the required invalidation semantics are met. Implementations that utilize a single Upstream TC are required to send a single Invalidate Completion in the utilized TC.

The ITag Vector field is used to indicate which Invalidate Request has been completed. Each of the 32 possible ITag field values from the Invalidate Request is represented by a single bit in the ITag Vector field. The least significant bit (bit 0; i.e., the right-most bit in the schematic representation of the Invalidate Completion message shown in § Figure 10-21 and § Figure 10-22) of the ITag Vector field corresponds to the ITag field value of 0. The most significant bit (bit 31) of the ITag Vector field corresponds to the ITag field value of 31. Implementations are allowed to coalesce multiple Invalidate Completions by setting multiple ITag Vector bits in a single message provided the following conditions are met:

- The Invalidate Completions flow in the same TC.
- The Invalidate Completions have the same CC value.
- All fragments of an Invalidate Completion must have identical Request ID, CC, and ITag Vector fields.

A TA that receives an Invalidate Completion for an ITag that has no outstanding Invalidate Request shall report this error using implementation specific mechanisms. One possible such mechanism is to report the Invalidate Completion as an Unexpected Completion (UC).

Functions that do not support ATS will treat an Invalidate Request as UR.

Functions supporting ATS are required to send an Invalidate Completion in response to a Invalidate Request independent of whether the Bus Master Enable bit is Set or not. Note that the above conditions must be satisfied even when Bus Master Enable is Cleared. The method for a device to achieve this is implementation dependent.

IMPLEMENTATION NOTE: BUS MASTER ENABLE CHANGE §

When Bus Master Enable changes from Set to Clear, no further memory requests should be queued. It is possible that queued write requests are present when BME is Cleared. These requests could block an Invalidate Completion. These requests must be either sent or dropped. This will ensure that all outstanding write transactions that are potentially dependent upon the outstanding invalidation are complete.

10.3.3 Invalidate Completion Semantics §

Before an ATC can return an Invalidate Completion for a given Invalidate Request, it must ensure the following conditions are satisfied with respect to stale addresses:

- All new Translated Memory Requests initiated by the Function (or its ~~↓↑Shadows~~)
~~↑↓Shadows, or its Scalable Device Interfaces~~)
will not utilize stale address translations.
- All expected Completions for Translated Memory Requests initiated by the Function (or its ~~↓↑Shadows~~)
~~↑↓Shadows, or its Scalable Device Interfaces~~)
and referencing stale addresses must be received.
- All expected Translation Completions for Translation Requests initiated by the Function (or its ~~↓↑Shadows~~)
~~↑↓Shadows, or its Scalable Device Interfaces~~)
and referencing stale addresses must be received or tagged to be discarded.
- All outstanding non-UIO Posted Translated Memory Requests initiated by the Function (or its ~~↓↑Shadows~~)
~~↑↓Shadows, or its Scalable Device Interfaces~~)
and referencing stale addresses must be ensured to be transmitted ahead of the associated Invalidate Completion(s).

ECN: Base 6.3
SIOV△↔

The ATC is required to send a copy of the Invalidate Completion message in each TC in which a non-UIO Posted Translated Memory Request has been issued but not known to have been pushed to observability on the fabric through the TA. The CC field must be set to the same value in each copy of the Invalidate Completion message indicating number of copies sent. The TA is responsible for collecting all sent responses before considering the invalidation to be complete. The ATC must send at least one Invalidate Completion message.

The ATC must not send copies of the Invalidate Completion message in UIO VCs (Posted Message TLPs are not defined for UIO VCs). When all traffic affected by an ATS Invalidate Request is UIO, it is implementation specific how the ATC picks a non-UIO TC to use for the Invalidate Completion message.

A TA must ensure forward progress of Requests without dependence on receiving Invalidate Completions.

Invalidate Completion Messages must contain the Requester ID used in the associated Invalidate Request.

When ~~↓Shadow Functions↓~~ ↑Shadow Functions↑ are enabled, it is sufficient to issue a single Invalidate Request message targeting either the Function or one of its Shadows. It is recommended that system software issue an Invalidate Request Message to the “main” Function instead of a Shadow Function. Since IDO is Clear, the Invalidate Completion resulting from that Invalidate Request will push to the TA any earlier Posted Writes for both the Function and its Shadows.

Invalidate Completion behavior is independent of whether the associated Invalidate Request was issued to the Function or one of its Shadows.

IMPLEMENTATION NOTE: IMPLIED TC FLUSHING §

When making the decision as to which TC to send Invalidate Completions, an ATC may infer, in an implementation specific manner, that an issued posted write has been pushed to the TA. For example, a Function that has sent a read transaction to a destination above the TA and received its corresponding response may infer that any preceding posted writes issued in the same TC have been pushed to the TA.

Independent of this optimization, the ATC is required to send at least one Invalidate Completion.

10.3.4 Request Acceptance Rules §

In accord with the request acceptance rules enumerated in this section, a Function is not allowed to create a dependency in which the acceptance of a posted transaction is dependent upon the transmission of a posted transaction. Given Invalidate Requests and Invalidate Completions both are posted transactions, Functions must not make the acceptance of an Invalidate Request dependent upon the transmission of an Invalidate Completion. The method for achieving this is implementation specific.

A Function with an ATS capability in its configuration space must be able to accept Invalidate Requests and send Invalidate Completions even if ATS is not enabled.

IMPLEMENTATION NOTE: INVALIDATE QUEUE DEPTH §

An ATC is only associated with a single TA. Each TA is limited to a total of 32 outstanding invalidations to any given ATC. This limits the number of outstanding Invalidate Requests active to a single ATC to 32. To avoid a post-to-post dependency, an ATC is required to accept up to 32 Invalidate Requests.

An ATC may choose to implement a maximally sized input queue holding Invalidate Requests. Alternatively, an ATC may choose to implement a maximally sized output queue holding Invalidate Completions. Note that queuing Invalidate Completions requires significantly less state per entry resulting in a potentially more efficient implementation than input queue buffering.

Note that the choice of whether to implement input queuing or output queuing (or a hybrid of both) has no impact on ensuring deadlock free behavior. But implementation choices with regard to queuing may have a significant impact on performance (see § Section 10.3.5).

10.3.5 Invalidate Flow Control §

Due to the variety of caching architectures and queuing strategies, implementations may vary greatly with respect to invalidation latency and throughput. It is possible that a TA may generate Invalidate Requests at a rate that exceeds the average ATC service rate. When this happens, the credit based flow control mechanisms will throttle the TA issue rate. A side effect of this is congestion spreading to other channels and Links through the credit based flow control mechanism. Depending on the frequency and duration of this congestion, performance may suffer. It is strongly recommended that TA and its associated software implement higher level flow control mechanisms.

To assist with the implementation of Invalidate Flow Control, an ATC must publish the number of Invalidate Requests it can buffer before back pressuring the Link. This field applies to all invalidations serviced by the Function, independent of the size of the invalidation. This value is communicated in the Invalidate Queue Depth field in the ATS capability structure (see § [Section 7.8.9](#)). A value of 0 0000b indicates that invalidate flow control is not necessary to this Function.

IMPLEMENTATION NOTE: INVALIDATE FLOW CONTROL §

A Function may indicate that invalidate flow control is not required when one or more of the following is true:

1. The Function can handle invalidations at the maximum arrival rate of Invalidate Requests.
2. The Function will not or very rarely cause Link backpressure (performance loss is negligible).
3. The Function can fully buffer the maximum number of incoming invalidations without back pressuring the Link.

Each Function is required to implement sufficient queuing to ensure it can hold the maximum number of outstanding Invalidate Requests from a TA (using either input or output queuing). However, with ~~↑↓an SR-IOV device, all VFs associated with a PF ↑↓share↑↓↑containing either an SR-IOV Extended Capability or an SIOV Extended Capability shares↑~~ a single Invalidate Request queue in the ~~↑↓PF.↓↑PF with all of its VFs or SDIs.↑~~ To implement invalidation flow control, the TA must ensure that the total number of outstanding Invalidate Requests targeting the shared PF queue does not exceed the value in the PF's Invalidate Queue Depth field.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

10.3.6 Invalidate Ordering Semantics §

Invalidate Requests and Translation Completions may be sent using different TC and are, therefore, unordered with respect to each other (from the Link's perspective). An ATC must ensure that the proper invalidation behavior is maintained when an Invalidate Request bypasses a Translation Completion to an overlapping region.

An ATC must “snoop” its outstanding translation request queue against all arriving Invalidate Requests. When snooping a request for a N*STU sized translation (N is a power of 2), the ATC must snoop the range of addresses starting at the STU aligned region containing the specified address and ending (N-1) STU size pages later.

If an Invalidate Request overlaps the address range in an outstanding Translation Request, the Translation Request must be tagged as invalid and the results of its corresponding Translation Response must be discarded prior to transmission of the Invalidate Completion. If the Translation Response is received before the Invalidate Completion is sent, an implementation is free to issue requests utilizing the translation result provided the Invalidate Completion Semantics (see § [Section 10.3.3](#)) are satisfied.

IMPLEMENTATION NOTE: REQUEST RANGE OVERLAP IN INVALIDATIONS §

In the description above, N is the number of STU sized translations that were requested in the Translation Request. This is equal to (Length field in Translation Request)/2.

As an example:

STU is 00 0010b indicating 16384-byte pages.

An outstanding Translation Request has a Length field of 00 0000 0100b indicating two translations covering a range of 32768 bytes.

The high-order 48 bits of the Translation Request are 0000 0FFF FFFFh.

The low-order 16 bits of the address in the request are 11xx xxxx xxxx xxxx b indicating that the translation request covers a range that overlaps a 32768-byte boundary (in fact, the request crosses a 16-TB boundary).

If two translations are returned, they would cover the two STU sized regions at 0000 0FFF FFFF C000h and 0000 1000 0000 0000h.

An Invalidate Request is received with the high-order 48 bits of 0000 1000 0000h and the low-order 16 bits of 0001 1xxx xxxx xxxx b.

The ATC must detect that a translation associated with a portion of the Translation Request is now invalidated and the Translation Completion associated with the invalidated region must be discarded (for simplification, the ATC is allowed to discard all of the Translation Completion).

It should be noted that, processing of the Invalidate Requests is simplified if Translation Requests do not cross alignment boundaries of the request. The Translation Request from the above example is not aligned to a 32768-byte boundary. If it were broken into two requests, it would be simpler to associate the range of the Invalidate Request with the address in the Translation Request. Breaking the Translation Requests into aligned requests is not a requirement.

10.3.7 Implicit Invalidations Events §

The following events will cause the invalidation of all ATC entries:

- Conventional Reset (all forms)
- Function Level Reset
- E field in ATS Capability changes from Clear to Set

The following events will cause the invalidation of all non-Global Mapping ATC entries that were requested using a specific PASID:

- Stopping the use of a PASID as defined in this specification.

No explicit Invalidate Completion message is sent when these implied invalidate events occur.

IMPLEMENTATION NOTE: IMPLICIT INVALIDATION AND PASID §

Software may not change any of the PASID enable bits when the E field in the ATS Capability is Set. The invalidation that occurs when software Sets the E field also invalidates ATC entries with an associated PASID value.

10.3.8 PASID and Global Invalidate §

The requirements in this section apply to Functions that support the PASID. For Invalidate Requests that have a PASID, the ATC shall:

- Optionally signal Unsupported Request (UR) if the associated PASID value is greater than or equal to $2^{\text{Max PASID Width}}$. This error may be signaled anytime an out of range PASID value is present, even when the PASID value is ignored (see below).
- Return an Invalidate Completion if PASID Enable is Clear.
- If the Function supports Global Invalidate (see § Section 7.8.9.2):
 - If the Global Invalidate bit in the Request is Set, invalidate Global and non-Global Mapping entries in the ATC within the indicated memory range associated with any PASID value and return an Invalidate Completion. The PASID value in the PASID is ignored.
 - If the Global Invalidate bit in the Request is Clear, invalidate only non-Global Mapping entries in the ATC within the indicated memory range that were requested using the associated PASID value and return an Invalidate Completion.

Global Mapping entries in the ATC for some or all of the indicated memory range may be retained.

- If the Function does not support Global Invalidate (see § Section 7.8.9.2), invalidate entries in the ATC within the indicated memory range that were requested using the associated PASID value and return an Invalidate Completion.
- If no matching entries are present in the ATC, invalidate no ATC entries and return an Invalidate Completion.

For Invalidate Requests that do not have a PASID, the ATC shall:

- Invalidate ATC entries within the indicate memory range that were requested without a PASID value.
- Invalidate ATC entries at all addresses that were requested with any PASID value.

10.4 Page Request Services §

The general model for a page request is as follows:

1. A Function determines that it requires access to a page for which an ATS translation is not available.
2. The Function causes the associated Page Request Interface to send a Page Request Message to its RC. A Page Request Message contains a page address and a Page Request Group (PRG) index. The PRG index is used to identify the transaction and is used to match requests with responses.

3. When the RC determines its response to the request (which will typically be to make the requested page resident), it sends a PRG Response Message back to the requesting Function.
4. The Function can then employ ATS to request a translation for the requested page(s).

A Page Request Message is a PCIe Message Request that is Routed to ~~↓↓the↓~~ Root Complex with a Message Code of 4 (0000 0100b). The mechanism employed at the RC to buffer requests is implementation specific. The only requirement is that an RC not silently discard requests.

All Page Request Messages and PRG Response Messages travel in PCIe Traffic Class 0. A Page Request Message or PRG Response Message with a Traffic Class other than 0 shall be treated as Malformed TLPs by the RC or endpoint that receives the same. Intermediate routing elements (e.g., Switches) shall not detect this error.

The Relaxed Ordering and ID-Based Ordering bits in the Attr field of Page Request Messages and PRG Response messages may be used. The No Snoop bit in the Attr field is reserved.

The page request service allows grouping of page requests into Page Request Groups (PRGs). A PRG can contain one or more page requests. All pages in a PRG are responded to en masse by the host. Individual pages within a PRG are requested with independent Page Request Messages and are recognized as belonging to a common PRG by sharing the same PRG index. The last request of a PRG is marked as such within its Page Request Message. One request credit is consumed per page request (not per PRG).

A PRG Response Message is a PCIe Message Request that is Routed by ID back to the requesting Function. It is used by system software to alert a Function that the page request(s) associated with the corresponding PRG has (have) been satisfied. The page request mechanism does not guarantee any request completion order and all requests are inherently independent of all other concurrently outstanding requests. If a Function requires that a particular request be completed before another request, the initial request will need to complete before the subsequent request is issued. It is valid for a Function to speculatively request a page without ascertaining its residence state and/or to issue multiple concurrently outstanding requests for the same page.

A Page Request Interface is allocated a specific number of page request message credits. An RC (system software) can divide the available credits in any manner deemed appropriate. Any measures the host chooses to employ to ensure that credits are correctly metered by Page Request Interfaces (a Page Request Interface is not using more than its allocation) is an implementation choice. A Page Request Interface is not allowed to oversubscribe the available number of requests (doing so can result in the page request mechanism being disabled if the buffer limit is exceeded at the root). A Page Request Interface's page request allocation is static. It is determined when the Page Request Interface is enabled and can only be changed by disabling and then re-enabling the interface.

10.4.1 Page Request Message §

A Function uses a Page Request Message to send page requests to its associated host. A page request indicates a page needed by the Function. The Page Request Interface associated with a Function is given a specific Page Request allocation. A Page Request Interface shall not issue page requests that exceed its page request allocation.

A page request contains the untranslated address of the page that is needed, the access permissions needed for that page, and a PRG index. A PRG Index is a 9-bit scalar that is assigned by the Function to identify the associated page request. Multiple pages may be requested using a single PRG index. When more than a single page is to be associated with a given PRG, the Last flag in the Page Request Record is cleared in all the requests except the last request associated with a given PRG (the flag is set in the last request). Page requests are responded to en masse. No response is possible (except for a Response Failure error) until the last request of a PRG has been received by the root. The number of PRGs that a Function can have outstanding at any given time is less than or equal to the associated Page Request Interface's Outstanding Page Request Allocation. It is valid for a request group to contain multiple requests for the same page and for multiple outstanding PRGs to request the same page.

A Page Request Interface applies to the “main” Function and its enabled ~~↓↓Shadow Functions↓~~ [↑↑Shadow Functions↑](#) (where the “main” is the Function that contains both the Page Request Extended Capability and the Shadow Function

Extended Capability). All Page Request Messages of a single PRG must have the same Requester ID. When **↓Shadow Functions** **↑Shadow Functions** are enabled, it is recommended that Page Request Messages use the Requester ID of the “main” Function.

The first two DWs of a Page Request Message contain a standard PCIe message header. The second two DWs of the message contain page request specific data fields.

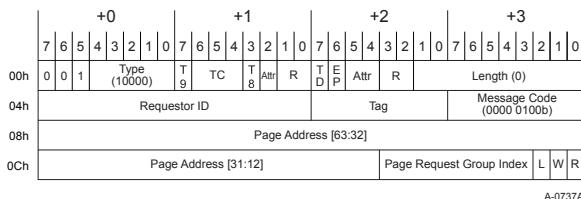


Figure 10-23 Page Request Message - Non-Flit Mode §

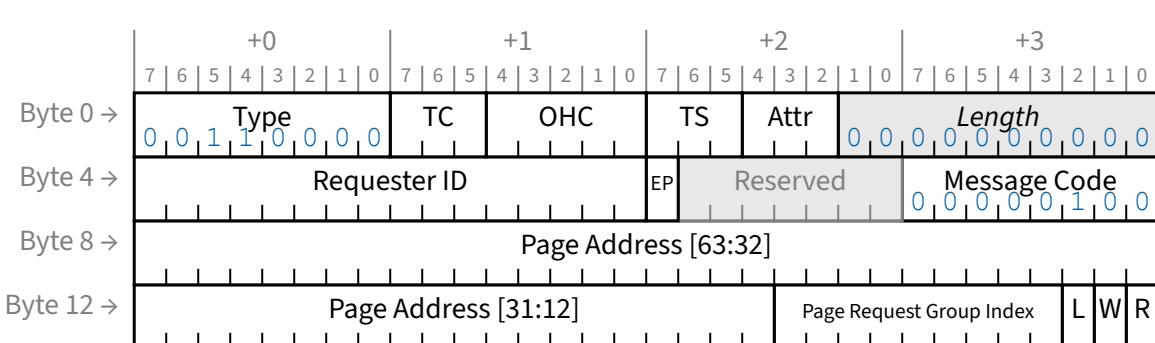


Figure 10-24 Page Request Message - Flit Mode §

Table 10-6 Page Request Message Data Fields §

| Field | Meaning |
|-------|---|
| R | <p>Read Access Requested - This field, when Set, indicates that the requesting Function seeks read access to the associated page. When Clear, this field indicates that the requesting Function will not read the associated page.</p> <p>The R field must be Set for Page Requests with a PASID and that have the Execute Requested bit Set.</p> <p>If R and W are both Clear and L is Set, this is a Stop Marker (see § Section 10.4.1.2.1).</p> |
| W | <p>Write Access Requested - This field, when Set, indicates that the requesting Function seeks write access and/or zero-length read access to the associated page. When Clear, this field indicates that the requesting Function will not write to the associated page.</p> <p>Upon receiving a Page Request Message with the W field Set, the host is permitted to mark the associated page dirty. Thus, Functions must not issue such Requests unless the Function has been given explicit write permission.</p> <p>If R and W are both Clear and L is Set, this is a Stop Marker (see § Section 10.4.1.2.1).</p> |
| L | <p>Last Request in PRG - This field, when Set, indicates that the associated page request is the last request of the associated PRG. A PRG can have a single entry, in which case the PRG consists of a single request in which this field is Set. When Clear, this field indicates that additional page requests will be posted using this record’s PRG Index.</p> |

| Field | Meaning |
|--------------------------|--|
| | If R and W are both Clear and L is Set, this is a Stop Marker (see § Section 10.4.1.2.1). |
| Page Request Group Index | Page Request Group Index - This field contains a Function supplied identifier for the associated page request. A Function need not employ the entire available range of PRG index values. A host shall never respond with a PRG Index that has not been previously issued by the Function and that is not currently an outstanding request PRG Index (except when issuing a Response Failure, in which case the host need not preserve the associated request's PRG Index value in the error response). |
| Page Address | Page Address - This field contains the untranslated address of the page to be loaded. For pages larger than 4096 bytes, the least significant bits of this field are ignored. For example, the least significant bit of this field is ignored when an 8096-byte page is being requested. |

IMPLEMENTATION NOTE: LAST BIT AND RELAXED ORDERING §

If multiple page requests are associated with a single PRG index, the last page request of a PRG should have the Relaxed Ordering attribute bit Clear in addition to having the Last flag Set. All other page request messages may have the Relaxed Ordering attribute bit set to any value.

10.4.1.1 PASID Usage §

The PASID Extended Capability indicates whether a Function supports PASID TLP Prefixes (NFM) or OHC-A1 with PASID (FM), and whether it is enabled to send and receive them.

Functions that support PASID are permitted to send a PASID on Page Request Messages. The PASID field contains the process address space of the page being requested and the Execute Requested and Privileged Mode Requested bits indicate the access being requested. In FM, the NW bit and the Last DW BE / 1st DW BE fields in the OHC-A1 are Reserved.

If one Page Request Message in a PRG has a PASID, all Page Request Messages in that PRG must contain identical PASID values. Behavior is undefined when the PASID values are inconsistent.

Functions that support PASID and have the PRG Response PASID Required bit Set (see § Section 10.5.2.3), expect that PRG Response Messages will contain a PASID if the associated Page Request Message had a PASID. For such PRG Response Messages, the Execute Requested and Privileged Mode Requested bits are reserved and the PASID field contains the PASID from the associated Page Request Message.

10.4.1.2 Managing PASID Usage on PRG Requests §

There are rules for stopping and starting the use of a PASID.

This section describes additional rules that apply to Functions that have issued Page Request Messages in a PASID that is being stopped. No additional rules are required to start the usage of the Page Request Interface for a PASID.

When stopping the use of a particular PASID, a Stop Marker Message may be optionally used to avoid waiting for PRG Response Messages before the Function indicates that the stop request for a particular PASID has completed.

To stop without using a Stop Marker Message, the Function shall:

1. Stop queueing new Page Request Messages for this PASID.
2. Finish transmitting any multi-page Page Request Messages for this PASID (i.e., send the Page Request Message with the L bit Set).
3. Wait for PRG Response Messages associated any outstanding Page Request Messages for the PASID.
4. Indicate that the PASID has stopped using a device specific mechanism. This mechanism must indicate that a Stop Marker Message will not be generated.

To stop with the use of a Stop Marker Message the Function shall:

1. Stop queueing new Page Request Messages for this PASID.
2. Finish transmitting any multi-page Page Request Messages for this PASID (i.e., send the Page Request Message with the L bit Set).
3. Internally mark all outstanding Page Request Messages for this PASID as stale. PRG Response Messages associated with these requests will return Page Request Allocation credits and PRG Index values but are otherwise ignored.²²⁰
4. Indicate that the PASID has stopped using a device specific mechanism. This mechanism must indicate that a Stop Marker Message will be generated.
5. Send a Stop Marker Message to indicate to the host that all subsequent Page Request Messages for this PASID are for a new use of the PASID value.

Note: Steps 4 and 5 may be performed in either order, or in parallel.

10.4.1.2.1 Stop Marker Messages §

A Stop Marker Message indicates that a Function has stopped using the Page Request Interface and has transmitted all pending Page Request Messages for a specific PASID. Stop Marker Messages are strongly ordered with respect to Page Request Messages and serve to push Page Request Messages toward the Host. When the Host receives the Stop Marker Message, this indicates that all Page Request Messages associated with the PASID being stopped have been delivered and that any subsequent Page Request Message with the same PASID value are associated with a new incarnation of that PASID value.

Stop Marker Messages do not have a response. They do not have a PRG Index and do not consume Page Request allocation (see § Section 10.5.2.5).

The Stop Marker Message bit layout is shown in § Figure 10-25 .

²²⁰. Page Request Allocation is shared across all PASIDs of the Function (see § Section 10.5.2.5). If PRG Response PASID Required is Clear, PRG Index values are shared across all PASIDs of the Function (see § Section 10.5.2.3).

| | | | | | | | | |
|-----|-------------------------------|-------------------------------|--------------|-------------------------------|----|-------------------------------|-------------------------|-------------------------------|
| | +0 | 7 6 5 4 3 2 1 0 | +1 | 7 6 5 4 3 2 1 0 | +2 | 7 6 5 4 3 2 1 0 | +3 | 7 6 5 4 3 2 1 0 |
| 00h | 1 0 0 1 0 0 0 1 | R R R | | | | PASID | | |
| 04h | 0 0 1 Type (10000) | T 9 | TC | T 8 Attr | R | T D E P Attr | R | Length (0) |
| 08h | | | Requester ID | | | Tag | | Message Code (0000 0100b) |
| 0Ch | | | | | | Reserved | | |
| 10h | | | | | | Reserved | Marker Type (00000b) | L W R 1 0 0 |

Figure 10-25 Stop Marker Message - Non-Flit Mode §

| | | | | | | | | |
|-----------|---------------------------------------|-------------------------------|-------------------|-------------------------------|----------|--|----------------------------------|-----------------------------------|
| | +0 | 7 6 5 4 3 2 1 0 | +1 | 7 6 5 4 3 2 1 0 | +2 | 7 6 5 4 3 2 1 0 | +3 | 7 6 5 4 3 2 1 0 |
| Byte 0 → | 0 0 1 Type 1 0 0 0 0 | TC | x x x x 1 | TS | Attr | 0 0 0 0 0 0 0 0 Length | | |
| Byte 4 → | | | Requester ID | EP | Reserved | | Message Code | 0 0 0 0 0 1 0 0 0 |
| Byte 8 → | | | | | | Reserved | | |
| Byte 12 → | | | | | | | Marker Type 0 0 0 0 0 | L W R 1 0 0 |

Figure 10-26 Stop Marker Message - Flit Mode §

A Stop Marker Message is encoded as a Page Request Message for which:

- In NFM, includes a PASID TLP Prefix. The Execute Requested and Privileged Mode Requested bits are Reserved.
- In FM, includes OHC-A1 with PASID. The Execute Requested bit, Privileged Mode Requested bit, and Last DW BE / 1st DW BE fields in the OHC-A1 are Reserved.
- The L, W and R fields contain 1b, 0b and 0b respectively.
- The Untranslated Address field and upper bits of the PRG Index field are Reserved.
- The Marker Type field contains 0 0000b to indicate that this is a Stop Marker Message.
- The Traffic Class must be 0.
- The Relaxed Ordering attribute bit must be Clear.
- The ID-Based Ordering attribute bit may be Set.

Behavior is undefined if a Stop Marker Message is received and any of the following are true:

- Marker Type not equal to 0 0000b.
- No PASID TLP Prefix is present (NFM).
- The PASID value does not match an outstanding stop request.
- An incomplete Page Request Message for the PASID is outstanding (i.e., for some PRG Index, the most recently received Page Request Message did not have the L bit Set).

10.4.2 Page Request Group Response Message §

System hardware and/or software communicate with a Function's page request interface via PRG Response Messages. A PRG Response Message is used by a host to signal the completion of a PRG, or the catastrophic failure of the interface. A single PRG Response Message is issued in response to a PRG, independent of the number of page requests associated with the PRG. There is no mechanism for indicating a partial request completion or partial request failure. If any of the pages associated with a given PRG cannot be satisfied, then the request is considered to have failed and the reason for the failure is supplied in the PRG Response Message. The host has no obligation to partially satisfy a multi-page request. If one of the requested pages cannot be made resident, then the entire request can, but need not, be discarded. That is, the residence of pages that share a PRG with a failed page request, but that are not associated with the failure, is indeterminate from the Function's perspective.

There are four possible Page Request failures:

1. The requested page is not a valid Untranslated Address.
2. PASID support exists, the Page Request has a PASID, and either PASID usage is not enabled for this request, the PASID value is not valid, or the Execute Requested bit is Set when R is Clear.²²¹
3. The requested page does not have the requested access attributes (including Execute permission and/or Privileged Mode access when those bits are present).
4. The system is, for an unspecified reason, unable to respond to the request. This response is terminal (the host may no longer respond to any page requests and may not supply any further replies to the Function until the Function's page request interface has been reset). For example, a request that violates a Function's assigned request limit or overflows the RC's buffering capability may cause this type of failure.

A Function's response to Page Request failure cases 1, 2, and 3 above is implementation dependent. The failure is not necessarily persistent, that is, a failed request may, in some instances succeed if re-issued. The range of possibilities precludes the precise specification of a generalized failure behavior, though on a per Function basis, the response to a failure will be an implementation dependent behavior.

All responses are sent to their associated Functions via PRG Response Messages. A Function must be capable of sinking multiple consecutive messages without losing any information. To avoid deadlock, a Function must be able to process PRG Response Messages for all of the Function's outstanding Page Request Messages without depending on the Function sending or receiving any other TLP.²²² A PRG Response Message is an ID routed PCIe message. The only Page Request Interface specific fields in this message are the Response Code and PRG. All other fields are standard PCIe message fields. (Note: these messages are routed based on the ID in bytes 8 and 9; with bytes 4 and 5 containing the host's Requester ID.)

Receipt of a PRG Response Message that contains a PRG Index that is not currently outstanding at a Function shall result in the UPRGI flag in the Page Request Extended Capability being Set, contingent upon the TLP otherwise being error free. Because of ambiguous language in earlier versions of this specification, it is permitted (though discouraged) to handle this case as an Unsupported Request (UR) or Unexpected Completion (UC) by the Function containing the Page Request Extended Capability, but otherwise no other error is permitted to be logged or signaled.

In order to prevent overflow, it is recommended to size Page Request queuing appropriately so that it does not overflow under expected behavior. If an overflow condition occurs, it is permitted to recover and resynchronize Page Request accounting as follows:

- The TA, upon detecting the overflow condition, stops accepting all incoming Page Requests for the queue and generates Page Request Group Responses with a Response Code of Success to Page Requests with L=1.

²²¹. Behavior when PASID support does not exist is defined in § Section 6.20.1.

²²². For example, processing a PRG Response Message that causes the Function to send a TLP Upstream must not block processing of subsequent downstream TLPs even if the Upstream TLP is delayed by flow control.

- Software processes all entries in the queue and generates Page Request Group Responses with a Response Code of Success to all Page Requests with L=1.
- Software clears the overflow condition to re-enable Page Request acceptance for the queue in the TA after all entries in the queue have been processed.

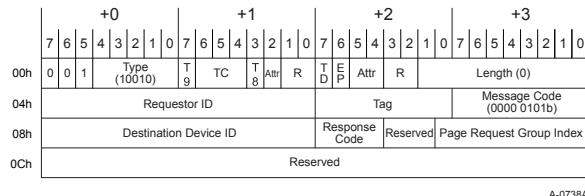


Figure 10-27 PRG Response Message - Non-Flit Mode §

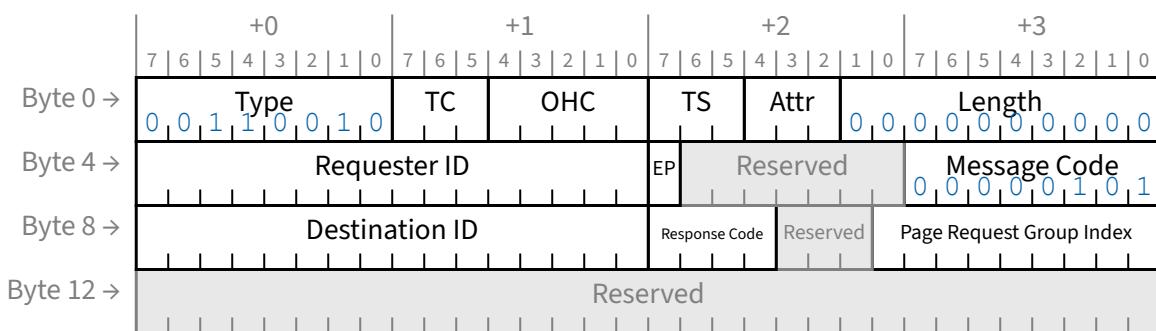


Figure 10-28 PRG Response Message - Flit Mode §

Table 10-7 PRG Response Message Data Fields §

| Field | Meaning |
|--------------------------|---|
| Page Request Group Index | Page Request Group Index - This field contains a Function supplied index to which the RC is responding. A given PRG Index will receive exactly one response per instance of PRG (with the possible exception of a Response Failure). |
| Response Code | Response Code - This field contains the response type of the associated PRG. The encodings are presented in § Section 10.4.2.1. |

10.4.2.1 Response Code Field §

The values and meaning for the Response Code field are listed in § Table 10-8 .

Table 10-8 Response Codes §

| Value | Status | Meaning |
|-------|-----------------|---|
| 0000b | Success | All pages within the associated PRG were successfully made resident. |
| 0001b | Invalid Request | One or more pages within the associated PRG do not exist or requests access privilege(s) that cannot be granted. Unless the page mapping associated with the Function is altered, re-issuance of the associated request will never result in success. |

| Value | Status | Meaning |
|-------------|------------------|--|
| 1110b:0010b | Unused | Unused Response Code values. A Function receiving such a message shall process it as if the message contained a Response Code of Response Failure. |
| 1111b | Response Failure | One or more pages within the associated request group have encountered/caused a catastrophic error. This response disables the Page Request Interface at the Function. Any pending page requests for other PRGs will be satisfied at the convenience of the host. The Function shall ignore any subsequent PRG Response Messages, pending re-enablement of the Page Request Interface. |

10.4.2.2 PASID Usage on PRG Responses §

If a Page Request has a PASID, the corresponding PRG Response Message may optionally contain one as well.

If the PRG Response PASID Required bit is Clear, PRG Response Messages do not have a PASID.

If the PRG Response PASID Required bit is Set, PRG Response Messages have a PASID if the Page Request also had one. The Function is permitted to use the PASID value from the prefix in conjunction with the PRG Index to match requests and responses.

When a PASID is attached to PRG Response Messages, the Execute Requested and Privileged Mode Requested bits are Reserved and the PASID value is copied from the PASID value of the Page Request.

10.5 ATS Configuration §

10.5.1 ATS Extended Capability §

Each Function that supports ATS (capable of generating Translation Requests) must have the ATS Extended Capability structure in its Extended Configuration Space. It is permitted to be implemented by Endpoint Functions or RCiEPs.

ATS support is optional in SR-IOV devices. If a VF implements an ATS capability, the ATS Extended Capability, its associated PF must implement an ATS capability. The ATS Capabilities in VFs and their associated PFs are permitted to be enabled independently.

ATS support is optional in SIOV Devices. If an SDI requires ATS functionality, its associated PF must implement the ATS Extended Capability. SDIs use the ATS settings from the ATS Extended Capability of their associated PF, however it is permitted for the SIOV Device to provide a device-specific mechanism to disable use of ATS on one or more SDIs while the ATS functionality is still enabled in the associated PF.

 ECN: Base 6.3
 SIOV△

Figure 10-29 details allocation of the register fields in the ATS Extended Capability structure.

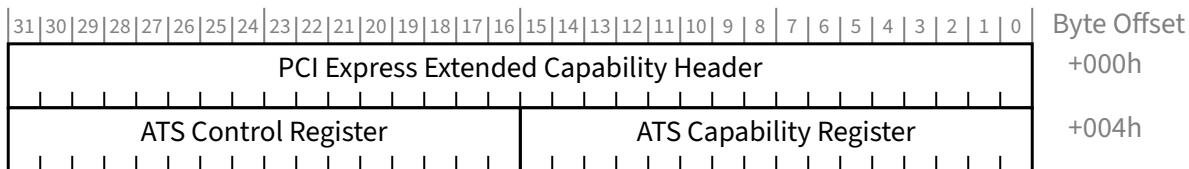


Figure 10-29 ATS Extended Capability Structure §

10.5.1.1 ATS Extended Capability Header (Offset 00h) §

§ Figure 10-30 details allocation of the register fields in the ATS Extended Capability Header ; § Table 10-9 provides the respective field definitions.

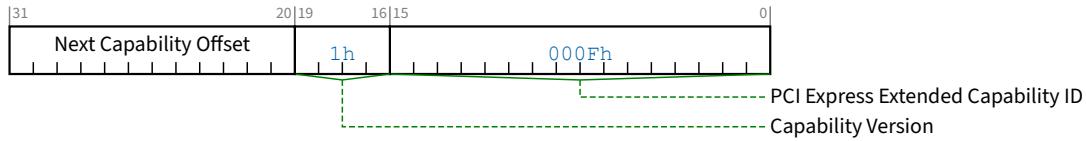


Figure 10-30 ATS Extended Capability Header §

Table 10-9 ATS Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - Indicates the ATS Extended Capability structure. This field must return a Capability ID of "000Fh" indicating that this is an ATS Extended Capability structure. | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be "1h" for this version of the specification. | RO |
| 31:20 | Next Capability Offset - The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

10.5.1.2 ATS Capability Register (Offset 04h) §

§ Figure 10-31 details the allocation of register fields of an ATS Capability Register ; § Table 10-10 provides the respective bit definitions.

Base 6.4 vs Base 6.3

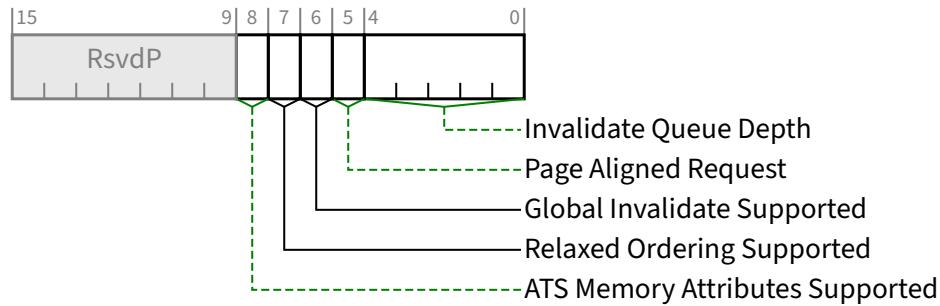


Figure 10-31 ATS Capability Register (Offset 04h) §

Table 10-10 ATS Capability Register (Offset 04h) §

| Bit Location | Register Description | Attributes |
|--------------|---|--------------|
| 4:0 | <p>Invalidate Queue Depth - The number of Invalidate Requests that the Function can accept before putting backpressure on the Upstream connection. A value of Zero in a non-VF Function indicates that it can accept 32 Invalidate Requests.</p> <p>For VFs, this field must be hardwired to Zero. VFs share the queue of the associated PF. See § Section 10.3.5 .</p> | RO VF ROZ |
| 5 | <p>Page Aligned Request - If Set, indicates the Untranslated Address is always aligned to a 4096 byte boundary. This field permits software to distinguish between implementations compatible with this specification and those compatible with an earlier version of this specification in which a Requester was permitted to supply anything in bits [11:2].</p> <p>It is strongly recommended that this bit be Set.</p> | RO |
| 6 | <p>Global Invalidate Supported - If Set, the Function supports Invalidate Requests that have the Global Invalidate bit Set. If Clear, the Function ignores the Global Invalidate bit in all Invalidate Requests (see § Section 10.3.8).</p> <p>It is strongly recommended that this bit be Clear.</p> <p>This bit must be hardwired to Zero if the Function does not support PASID.</p> <p style="background-color: yellow;">↑↑This bit must be hardwired to Zero if the TEE-IO Supported bit is Set in the Device Capabilities Register .↑</p> | RO |
| 7 | <p>Relaxed Ordering Supported - If Set, indicates this Function is permitted to Set the RO bit in Translation Requests when Enable Relaxed Ordering bit is Set.</p> | RO |
| 8 | <p>ATS Memory Attributes Supported - If Set, the Function supports using AMA values from ATS Translation Completions in the Function ATC. When Clear, the Function ignores the AMA field in ATS Translation Completions.</p> | RO |
| 9 | <p style="background-color: yellow;">↑↑ECN: Base 6.3 XT△↔↑</p> <p style="background-color: yellow;">↑↑TEE Exclusive Memory Attribute Supported - If Set, the Function supports use of TEE Exclusive Memory Attribute from Translation Completions. If Clear, the Function ignores the TEE Exclusive Memory Attribute in all Translation Completions (see § Section 10.2.3.10).↑</p> <p style="background-color: yellow;">↑↑This bit must be hardwired to Zero if the Global Invalidate Supported bit is Set.↑</p> <p style="background-color: yellow;">↑↑This bit must be hardwired to Zero if the TEE-IO Supported bit is Clear in the Device Capabilities Register.↑</p> | ↑↑RO↑ |

10.5.1.3 ATS Control Register (Offset 06h) §

§ Figure 10-32 details the allocation of register fields of an ATS Control Register ; § Table 10-11 provides the respective bit definitions.

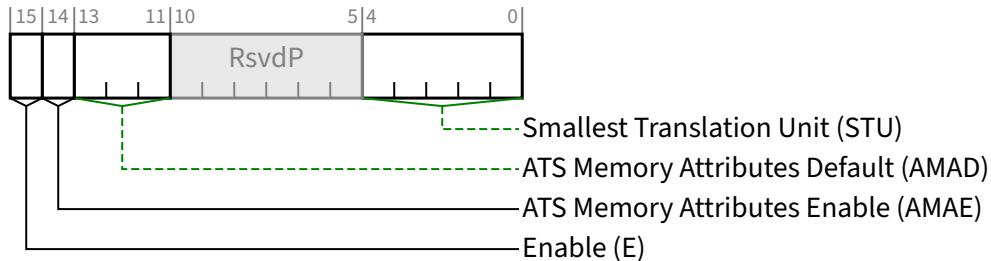


Figure 10-32 ATS Control Register §

Table 10-11 ATS Control Register §

| Bit Location | Register Description | Attributes | | | | |
|--------------|--|--------------|-----------------------|-----------|------------------------------|---------|
| 4:0 | <p>Smallest Translation Unit (STU) - This value indicates to the Function the minimum number of 4096-byte blocks that is indicated in a Translation Completions or Invalidate Requests. This is a power of 2 multiplier and the number of blocks is 2^{STU}. A value of 0 0000b indicates one block and a value of 1 1111b indicates 2^{31} blocks (or 8 TB total)</p> <p>For VFs, this field must be hardwired to Zero. The associated PF's value applies.</p> <p>Default value is 0 0000b.</p> | RW VF ROZ | | | | |
| 13:11 | <p>ATS Memory Attributes Default (AMAD) – When Set, as a performance optimization, and when AMAE is Set, the Requester is permitted to provide only non-default AMA values in Requests using the TPH TLP Prefix.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>AMAD</td> <td>ATS Memory Attributes</td> </tr> <tr> <td>000b-111b</td> <td>ATS Memory Attribute values.</td> </tr> </table> <p>This field is permitted to be hardwired to 000b if ATS Memory Attributes Supported is Clear.</p> <p>Default value is 000b.</p> | AMAD | ATS Memory Attributes | 000b-111b | ATS Memory Attribute values. | RW / RO |
| AMAD | ATS Memory Attributes | | | | | |
| 000b-111b | ATS Memory Attribute values. | | | | | |
| 14 | <p>ATS Memory Attributes Enable (AMAE) - When Set, the Requester is permitted to provide AMA values in Requests using the TPH TLP Prefix.</p> <p>This bit is permitted to be hardwired to 0b if the ATS Memory Attributes Supported bit is Clear.</p> <p>Default value is 0b.</p> | RW / RO | | | | |
| 15 | <p>Enable (E) - When Set, the Function is enabled to cache translations.</p> <p>Behavior is undefined if this Endpoint Function supports PASID, this bit is Set and any bit in the associated PASID Control Register is changed (see § Section 7.8.9.3)</p> <p>Default value is 0b.</p> | RW | | | | |

Base 6.4 vs Base 6.3

10.5.2 Page Request Extended Capability Structure §

A Page Request Extended Capability Structure is used to configure the Page Request Interface mechanism. A Multi-Function Device or RCiEP Device may implement a Page Request Interface and the associated capability on any Endpoint Function within the Device. **Every Page Request Interface mechanism operates independently.** For **SR-IOV devices**, a single Page Request Interface is permitted for the PF and is shared between the PF and its associated VFs, in which case the PF implements this capability and its VFs must not. **For SIOV Devices , a single Page Request Interface is permitted for the PF and is shared between the PF and its associated SDIs . If an SDI requires PRI functionality, its associated PF must implement the Page Request Interface and the associated capability. SDIs use the PRI settings from the Page Request Extended Capability of their associated PF, however it is permitted for the SIOV Device to provide a device-specific mechanism to disable use of PRI on one or more SDIs while the PRI functionality is still enabled in the associated PF.**

ECN: Base 6.3
SIOV△↔

For an **Endpoint employing SR-IOV device, or SIOV**, even though the Page Request Interface is shared between its PFs and VFs, it sends **identifies** the **requesting Function's ID (PF, VF, or VF) in the SDI via the Requester ID field of the Page Request Message, and the requesting Function's same ID must be in the Destination Device ID field of the resulting PRG Response Message.**

ECN: Base 6.3
SIOV△↔

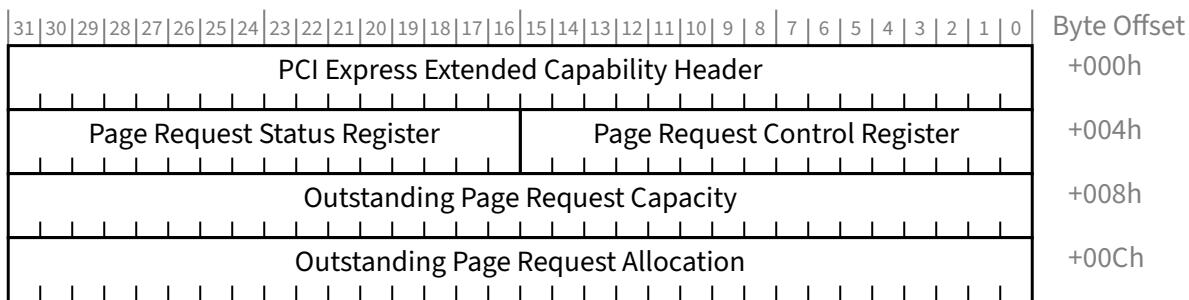


Figure 10-33 Page Request Extended Capability Structure §

10.5.2.1 Page Request Extended Capability Header (Offset 00h) §

Figure 10-34 details allocation of the register fields in the Page Request Extended Capability Header ; Table 10-13 provides the respective field definitions.

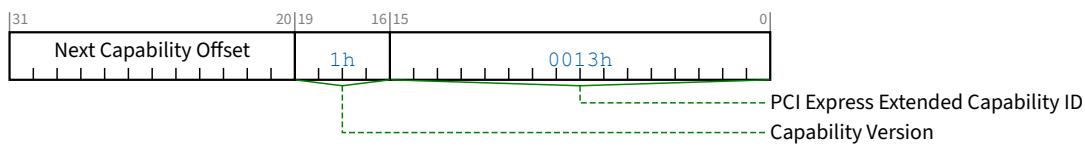


Figure 10-34 Page Request Extended Capability Header §

Table 10-13 Page Request Extended Capability Header §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | PCI Express Extended Capability ID - Indicates that the associated extended capability structure is a Page Request Extended Capability. This field must return a Capability ID of "0013h". | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be "1h" for this version of the specification. | RO |
| 31:20 | Next Capability Offset - The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

10.5.2.2 Page Request Control Register (Offset 04h) §

§ Figure 10-35 details allocation of the register fields in the Page Request Control Register ; § Table 10-14 provides the respective field definitions.

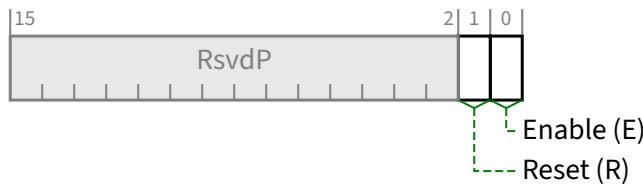


Figure 10-35 Page Request Control Register §

Table 10-14 Page Request Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | Enable (E) - This field, when set, indicates that the Page Request Interface is allowed to make page requests. If this field is Clear, the Page Request Interface is not allowed to issue page requests. If both this field and the Stopped field are Clear, then the Page Request Interface will not issue new page requests, but has outstanding page requests that have been transmitted or are queued for transmission. When the Page Request Interface is transitioned from not-Enabled to Enabled, its status flags (Stopped, Response Failure, and Unexpected Page Request Group Index (UPRGI) flags) are cleared. Enabling a Page Request Interface that has not successfully Stopped has indeterminate results. Default value is 0b. | RW |
| 1 | Reset (R) - When the Enable field is clear, or is being cleared in the same register update that sets this field, writing a 1b to this field, clears the associated implementation dependent page request credit counter and pending request state for the associated Page Request Interface. No action is initiated if this field is written to 0b or if this field is written with any value while the Enable field is Set. Reads of this field return 0b | RW |

10.5.2.3 Page Request Status Register (Offset 06h) §

§ Figure 10-36 details allocation of the register fields in the Page Request Error Register; § Table 10-15 provides the respective field definitions.

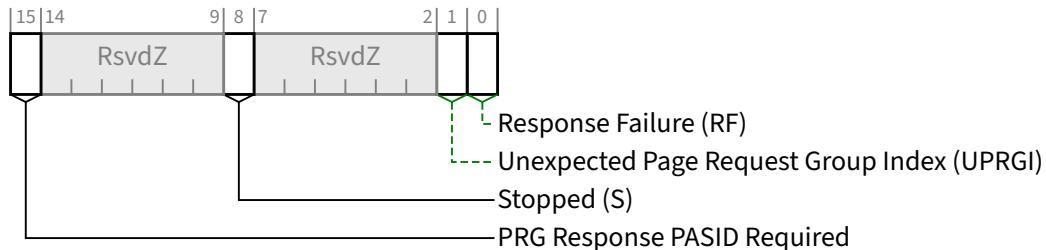


Figure 10-36 Page Request Status Register §

Table 10-15 Page Request Status Register §

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 0 | <p>Response Failure (RF) - This field, when Set, indicates that the Function has received a PRG Response Message indicating a Response Failure. The Function expects no further responses from the host (any received are ignored). This field is Set by the Function and Cleared when a one is written to the field.</p> <p>For ↓↓SR-IOV, ↓↑SR-IOV or SIOV, ↑ this field is Set in the PF if ↓↑the PF or↑ any ↓↑of its↑ associated ↓↓Function (PF or VF)↓ ↓↑VF/ SDI sharing the Page Request Interface↓ receives a PRG Response Message indicating Response Failure.</p> <p>Default value is 0b.</p> | RW1C |
| 1 | <p>Unexpected Page Request Group Index (UPRGI) - This field, when Set, indicates that the Function has received a PRG Response Message containing a PRG index that has no matching request. This field is Set by the Function and cleared when a one is written to the field.</p> <p>For ↓↓SR-IOV, ↓↑SR-IOV or SIOV, ↑ this field is Set in the PF if ↓↑the PF or↑ any ↓↑of its↑ associated ↓↓Function (PF or VF)↓ ↓↑VF/ SDI sharing the Page Request Interface↓ receives a PRG Response Message that ↓↓does↓ has no matching request.</p> <p>Default value is 0b.</p> | RW1C |
| 8 | <p>Stopped (S) - When this field is Set, the associated page request interface has stopped issuing additional page requests and all previously issued Page Requests have completed. When this field is Clear the associated page request interface either has not stopped or has stopped issuing new Page Requests but has outstanding Page Requests. This field is only meaningful if Enable is Clear. If Enable is Set, this field is undefined.</p> <p>When the Enable field is Cleared, after having been previously Set, the interface transitions to the stopping state and Clears this field. After all page requests currently outstanding at the Function(s) have completed, this field is Set and the interface enters the disabled state. If there were no outstanding page requests, this field may be Set immediately when Enable is Cleared. Resetting the interface will cause an immediate transition to the disabled state. While in the stopping state, receipt of a Response Failure message will result in the immediate transition to the disabled state (Setting this field).</p> | RO |

| Bit Location | Register Description | Attributes |
|--------------|--|----------------------|
| | <p>For ↓↓SR-IOV, ↑↑SR-IOV or SIOV, ↑ this field is Set ↑↑in the PF only when ↑↑the PF and ↑ all associated ↑↑Functions (PF and VFs) ↓↑VFs/ SDIs ↑ have stopped issuing page requests.</p> <p>Default value is 1b.</p> | ECN: Base 6.3 SIOV△↔ |
| 15 | <p>PRG Response PASID Required - If Set, the Function expects a PASID on PRG Response Messages when the corresponding Page Requests had a PASID. If Clear, the Function does not expect PASID on any PRG Response Message.</p> <p>Function behavior is undefined if this bit is Clear and the Function receives a PRG Response Message with a PASID.</p> <p>Function behavior is undefined if this bit is Set and the Function receives a PRG Response Message with no PASID when the corresponding Page Requests had a PASID.</p> <p>This bit is RsvdZ if the Function does not support PASID.</p> | RO |

10.5.2.4 Outstanding Page Request Capacity (Offset 08h) §

This register contains the number of outstanding page request messages the associated Page Request Interface physically supports. This is the upper limit on the number of pages that can be usefully allocated to the Page Request Interface.

This register is Read Only.

10.5.2.5 Outstanding Page Request Allocation (Offset 0Ch) §

This register contains the number of outstanding page request messages the associated Page Request Interface is allowed to issue (have outstanding at any given instance).

The number of PRGs a Page Request Interface has outstanding is less than or equal to the number of request messages it has issued. For example, if system software allocates 1000 messages to a Page Request Interface then a single PRG could use all 1000 of the possible requests. Conversely, at one request per PRG the Page Request Interface would run out of PRG indices (of which there are only 512) before it consumes all its page request credits. A Page Request Interface must pre-allocate its request availability for any given PRG, that is, all the requests required by a given PRG must be available before any of the requests may be issued.

When ~~↓↓Shadow functions↓↑Shadow Functions↑~~ are enabled, this allocation applies to the Function containing this capability and its enabled ~~↓↓Shadow Functions↓↑Shadow Functions↑~~ (See § Section 7.9.25).

This register is Read/Write. Behavior is undefined if this register is changed while the Enable flag is set. Behavior is undefined if this register is written with a value larger than Outstanding Page Request Capacity . Default value is 0.

When PASID is supported, the Request Allocation remains associated with the Function and is shared across the Function as well as all PASIDs of the Function.

Stopping a PASID does not affect any allocation used by that PASID. The system should continue to respond with PRG Response Messages in order to return Page Request and PRG Index resources to the Function (see § Section 10.4.2.1).

Stop Marker Messages consume buffering but are not included in this allocation (see § Section 10.4.1.2.1). Systems should provide additional buffering for Stop Marker Messages and should limit the number of outstanding Stop Marker Messages to avoid overrunning this additional buffering.

Base 6.4 vs Base 6.3

11. TEE Device Interface Security Protocol (TDISP) §

Trusted Execution Environments (TEEs) that include a composition of resources from one or more devices and the host require mechanisms to establish and manage trust relationships. Here we will use the term TEE-I/O to refer to a conceptual framework for performing such operations. This chapter defines a specific architecture for hosts and devices to participate in TEE-I/O (see § Figure 11-1).

TEE-I/O builds upon existing capabilities for the direct assignment of devices to VMs, such as SR-IOV [↑↑and SIOV↑](#) ([↑↑\\$ Chapter 9.↑](#)) and ATS (§ Chapter 10.), to establish Trusted Execution Environment VMs (TVMs). All VMs that are not TVMs are referred to as legacy VMs. In TEE-I/O, the VMM itself may not be trusted by TVMs , and mechanisms are provided to enable the TVM to make trust decisions based on the underlying hardware it is using. Although the VMM is not required to be trusted by TVMs , it continues to perform the resource allocation and system management functions as it does in non-TEE-I/O use models, but in such a way that the results can be tested. The VMM can be blocked from bypassing the security of the affected TVM (s). Legacy VMs that implicitly trust the VMM may co-exist with TVMs in a system.

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

Base 6.4 vs Base 6.3

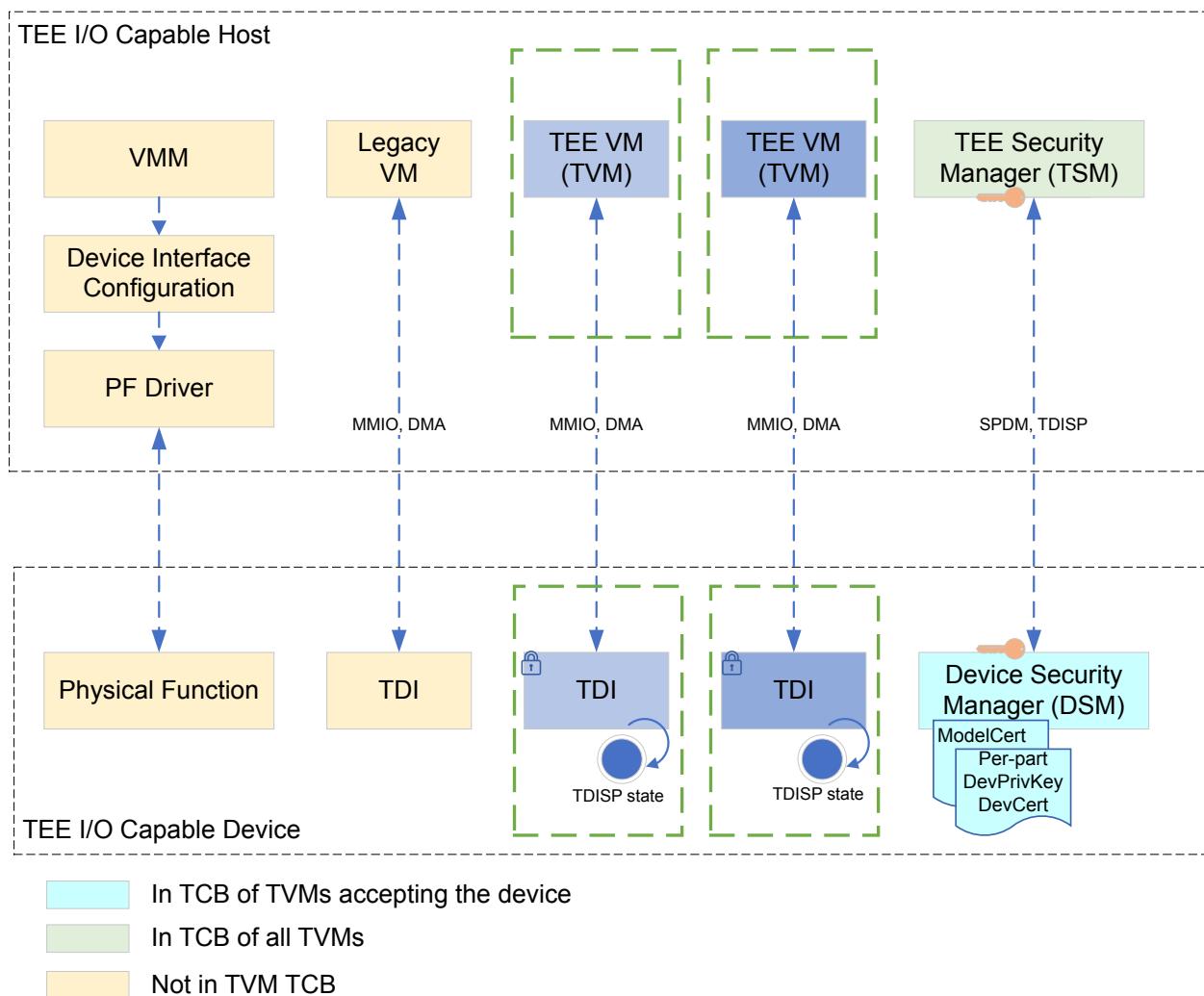


Figure 11-1 Conceptual View with Example Host and Device and Logical Communication Paths §

The **TEE Security Manager (TSM)** is a logical entity in a host that is in the TCB for a TVM and enforces security policies on the host. The Device Security Manager (DSM) is a logical entity in the device that may be admitted into the TCB for a TVM by the TSM and enforces security policies on the device.

The TEE Device Interface (TDI) Security Protocol (TDISP) defines an architecture for devices that support TEE-I/O virtualization, providing the following functions:

1. Establishing a trust relationship between a TVM and a device.
2. Securing the interconnect between the host and device.
3. Attach and detach a TDI to a TVM in a trusted manner.

Although TDISP has been defined in relation to TEE-I/O as described above, TDISP stands alone as a specification for devices, and such devices may be operated in systems using security architectures other than TEE-I/O, provided that the host functions required by TDISP are supported in an appropriate way by the system.

TDISP defines requirements for TDIs specifically, and also for the entire Device implementing TDIs , where in a specific instance, a TDI may be an entire Device, a non-IOV Function, a PF (and possibly its subordinate ~~↓↑VFs~~,~~↑↑VFs~~ or ~~↑↑SDIs~~,~~↑~~ a ~~↓↑VF~~,~~↑↑VF~~, or an SDI .~~↑~~ Although it is permitted (and generally expected) that TDIs will be implemented such that they can be assigned to Legacy VMs, such use is not the focus of TDISP .

ECN: Base 6.3
SIOV $\triangleleft\triangleright$

11.1 Overview of the TEE-I/O Security Model as it Relates to Devices §

The TEE-I/O security model is primarily intended to apply to systems using device resources directly assigned to VMs, and this chapter generally assumes this use case. However, devices that are compliant to TDISP can potentially be used in other ways, and such use is not prohibited, although it is outside the scope of this specification.

The TEE-I/O considers all resources, including memory of all types, host processors, TDI's and, in some cases internal state, to be in one of two classes:

- “TEE-assignable” resources have the required trust/security capabilities to be assigned to a TEE. Once assigned to a TEE, these resources become “TEE-Owned.”
- “Non-TEE-assignable” resources either do not possess the required trust/security capabilities to be assigned to a TEE or have been excluded by some ~~↓↑implementation-specific~~,~~↑↑implementation specific~~ mechanism. These resources may, and often do, have a critical role in system function, and therefore it is often desirable to appropriately secure them, although how this is done is outside the scope of this specification.

The TEE-I/O security model does not require the VMM to be trusted by TVMs. Therefore, devices supporting hardware-assisted I/O virtualization (e.g., SR-IOV) require security extensions to ensure that the device’s virtualization model does not allow or require intervention by software outside the TVM trust boundary to perform operations that affect the confidentiality and/or integrity of TVM data in-flight or at-rest in the device. The primary focus of TDISP is to define the requirements for TDISP-compliant devices and the necessary elements outside of such devices required to support the TDISP architecture. Additional system capabilities required are outside the scope of TDISP.

TVM data, code, and execution state stored in an assigned device must be protected against:

- Confidentiality breaches: read access by entities (firmware, software, or hardware) not in the TCB of the TVM (such as other TVMs, VMM, etc.).
- Integrity: modification by entities (firmware, software, or hardware) not in the TCB of the TVM (such as other TVMs, VMM, etc.).

This security model does not require protection of TVMs against denial-of-service attacks in general. However, systems may impose a requirement that a TVM not have the ability to cause denial of service to other TVMs, VMM, or other VMs executing on the platform. The TSM by itself may not have all the capabilities needed to defend the platform against denial of service. Enforcing this property is the collective responsibility of the TSM , VMM, device and DSM , and is outside the scope of this specification.

The hardware assisted I/O virtualization schemes for direct I/O from TVMs to devices must address the following to preserve the confidentiality and integrity of the TVMs and the data moved between the TVMs and devices:

1. Authenticating device identity and measurement reporting - Device identities like Vendor ID and Device ID may be spoofed with malicious intent. Firmware executing on devices may have security vulnerabilities, or may have been tampered with. Device debug interfaces may be used to obtain low level access to the device hardware and thereby influence the security property of devices. The TVM must be able to cryptographically check the identity of the device, identity of the firmware components running on the device, and security state of the device (e.g., debug active). CMA/SPDM is used to support these requirements.

2. Device to Host communication Security - Physical access may be used to tamper with the data transferred between the host and the device. Transfers must be cryptographically protected to provide confidentiality, integrity, and replay protection to TVM data, and such schemes must also guard against violations of producer-consumer ordering. IDE is used to support these requirements. In some cases, e.g. for an RCiEP, it may be possible to ensure by construction that communication is not be susceptible to tampering, and therefore may not require the use of IDE. The TSM and DSM are both responsible for ensuring that Device/Host (and, when peer-to-peer is used, Device/Device) communication is secured by IDE, or by other means that satisfy use model requirements.
3. TEE Device Interface (TDI) management - DMA and interrupt remapping tables set up by the VMM may be tampered with by the VMM. The VMM administration of these tables (e.g., IOMMU TLB management, Device TLB management, Page Request handling, etc.) may additionally be tampered with by the VMM to influence the security of the TVM interaction with the device. The device must support locking down configurations of the TDI, reporting the configurations in a trusted manner, securely placing the TDIs into operational state, and subsequently tearing them down when the TDI is detached from a TVM . This chapter defines the mechanisms used to manage the security states of TDIs.
4. Device Security Architecture - Administrative interfaces (e.g., a PF) may be used to influence the security properties of the TDI used by the TVM . The device's security architecture must provide isolation and access control for TVM data in the device for protection against entities that are not in the trust boundary of the TVM . This chapter defines some device security architecture requirements, but additional requirements may exist for specific implementations that are outside the scope of this specification.

This chapter defines the wire protocol and the security objectives that are required to be implemented by the host and the device to be compatible with the TEE-I/O framework and the capabilities that need to be implemented to achieve specified security objectives. The implementation of such capabilities and the physical manifestation of the logical entities are outside the scope of this specification.

Base 6.4 vs Base 6.3

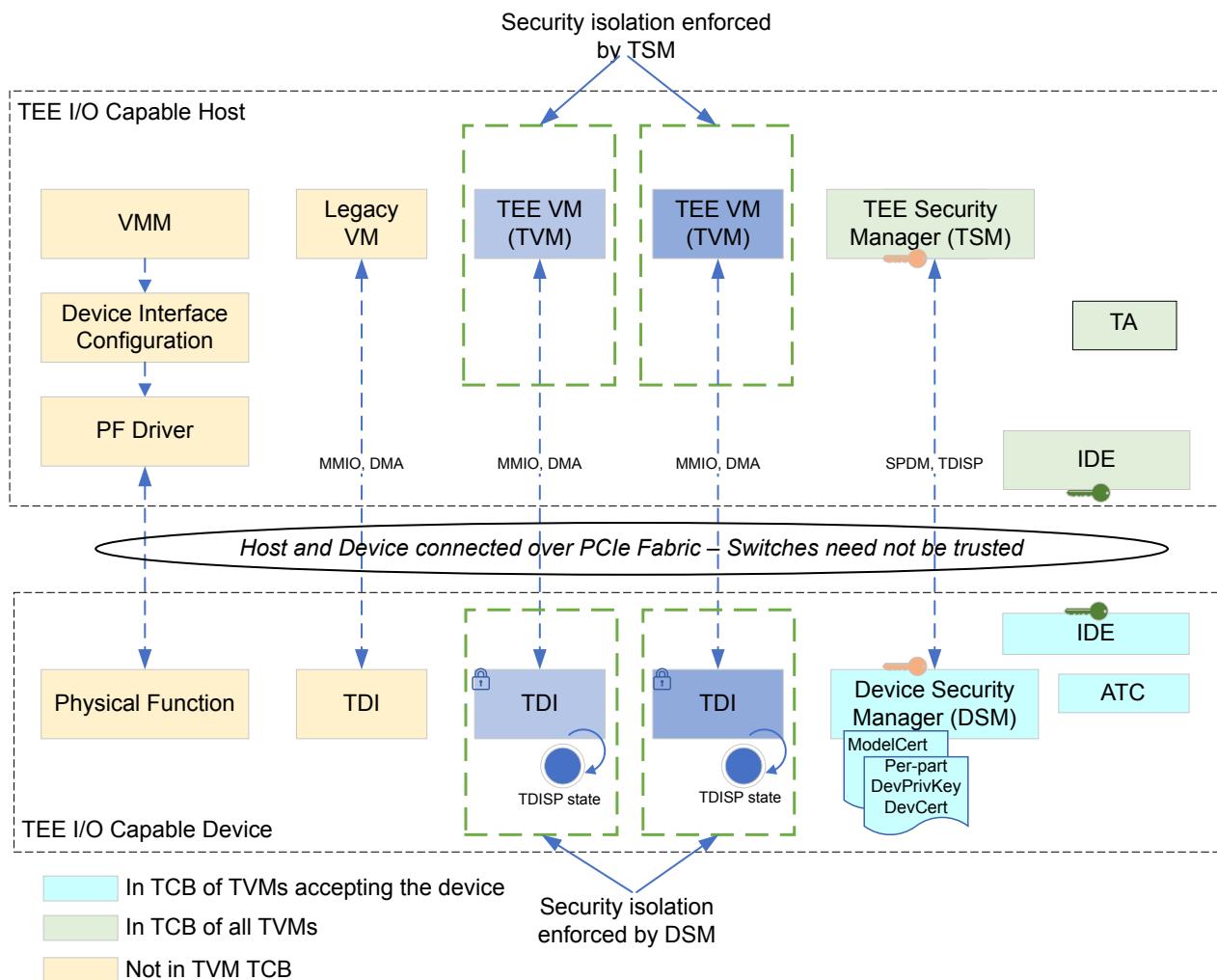


Figure 11-2 TDSP Host/Device Reference Architecture §

§ Figure 11-2 illustrates key elements of the TDSP reference architecture. Typically, a PF is the resource management entity for a TDI and is managed by the PF driver in the VMM. The VMM and the PF driver are not required to be in the TCB of the TVMs. The VMM uses the PF to configure TDIs for assignment to TVMs. TEE-I/O requires the device to organize its hardware/software interfaces such that the PF cannot be used to affect the security of a TDI when it is in use by a TVM. The device must support mechanisms to lockdown the configurations of the TDIs, when requested by the TSM, such that any modifications to the TDI configurations, once the TVM has accepted and started using the TDI, are detected as malicious actions. The device is required to implement a security architecture that protects the confidentiality and integrity of TVM data from being tampered with by the PF or other TDIs assigned to other TVMs or VMs. To ensure that error conditions can be appropriately managed, the device should implement Advanced Error Reporting (AER).

There are a variety of additional elements of the reference architecture. Software running on a Host CPU must be associated with a TEE via implementation-specific means. Memory can be a system-level resource or TEEs are associated with a TDI, and is defined as either two types of memory: TEE memory or non-TEE memory. TEE memory must include the memory used for the TEE's code, data, and execution state. Additionally, the memory in TEE Device Interfaces (TDIs) that are associated with the TEE and have mechanisms the IS_NON_TEE_MEM attribute Cleared (see § Section 11.3.22) also falls ECN: Base 6.3 XT

under the TEE memory. All other memory related to $\uparrow\downarrow$ ensure $\uparrow\downarrow$ the $\uparrow\downarrow$ TEE is classified as non-TEE memory. TEE memory is protected for confidentiality $\uparrow\downarrow$ and integrity by restricting access only to entities within the TEE's TCB. Conversely, non-TEE memory is accessible to entities outside the TEE's TCB, and its contents are not considered trustworthy by the TEE. \uparrow

$\uparrow\downarrow$ The eXtended TEE (XT) Mode allows the originator of $\uparrow\downarrow$ TVM data, $\uparrow\downarrow$ a TLP to indicate whether a TLP is associated with a TEE, much like what was possible without the XT Mode. Additionally, for TEE-originated Requests, the XT Mode enables specification of whether the translation agent (TA) and/or the Completer should restrict the requested access to either TEE memory or non-TEE memory, or allow access to either when the distinction is immaterial. Requests not originating from a TEE are restricted to accessing only non-TEE memory. \uparrow

$\uparrow\downarrow$ The XT bit \uparrow and $\uparrow\downarrow$ may additionally \downarrow $\uparrow\downarrow$ the T bit are used by the device and the TA to provide $\uparrow\downarrow$ integrity properties on \downarrow $\uparrow\downarrow$ access control to TVM assigned memory and memory mapped I/O registers. The XT bit and \uparrow the $\uparrow\downarrow$ T bit must be used only as defined in $\uparrow\downarrow$ TVM \downarrow $\uparrow\downarrow$ \$ Table 11-1 \uparrow $\uparrow\downarrow$ data. \downarrow $\uparrow\downarrow$.

ECN: Base 6.3 XT $\triangle\Delta$ ECN: Base 6.3 XT $\triangle\Delta$ Table 11-1 $\uparrow\downarrow$ XT Bit and T Bit Encodings \downarrow ECN: Base 6.3 XT $\triangle\Delta$

| $\uparrow\downarrow$ XT \uparrow | $\uparrow\downarrow$ T \uparrow | $\uparrow\downarrow$ Meaning of Encoding \uparrow |
|------------------------------------|-----------------------------------|---|
| $\uparrow\downarrow$ 0 \uparrow | $\uparrow\downarrow$ 0 \uparrow | Non-TEE $\uparrow\downarrow$ originator \uparrow |
| $\uparrow\downarrow$ 0 \uparrow | $\uparrow\downarrow$ 1 \uparrow | TEE originator \uparrow |
| $\uparrow\downarrow$ 1 \uparrow | $\uparrow\downarrow$ 0 \uparrow | TEE originator, the Request must be restricted to accessing non-TEE \uparrow memory $\uparrow\downarrow$ is not assumed \downarrow |
| $\uparrow\downarrow$ 1 \uparrow | $\uparrow\downarrow$ 1 \uparrow | TEE originator, the Request must be restricted \downarrow to $\uparrow\downarrow$ have any such mechanisms. \downarrow $\uparrow\downarrow$ accessing TEE memory \uparrow |

System configuration of Memory Address routing mechanisms must be managed so as to ensure correct system operation, as misrouting of TLPs will in many cases result in conditions indistinguishable from an attack, in turn resulting in an error condition, such as a Misrouted IDE TLP error or an IDE Check Failed error. Except when a peer-to-peer connection has been established between two TDIs, all Requests must be routed to the Root Complex, and in some cases this result is achieved by means of Access Control Services (see § Section 6.12) mechanisms that modify the routing of TLPs.

When Links that could be subject to physical attacks are used, Integrity and Data Encryption (IDE) must be supported and enabled. The use of Selective IDE Streams minimizes the TCB and attack surface by allowing intermediate Switches to be excluded from the TCB. For Endpoint Upstream Ports connected directly to Root Ports, Link IDE meets the stated requirement of minimizing TCB and attack surface, and it is acceptable in such configurations to use Link IDE instead of Selective IDE, provided the TSM and DSM are able to provide acceptable security in this configuration. It is permitted for IDE streams established by the TSM to be used to carry TLPs associated with legacy VMs. Between the same two Ports, separate IDE streams per TDI do not provide additional protection against an adversary employing physical attacks on a Link, so only a single IDE Stream is required. The TLPs once decrypted and authenticated at the device or at the Root Port are in cleartext, and access control mechanisms put in place by the TSM on the host, and the DSM on the device, must provide confidentiality and integrity to the TLP contents against entities not in the TCB of the TVMs. How this is done is outside the scope of this specification.

An RCiEP or other TDI integrated into a Root Complex may not require use of IDE to protect the TLPs, and as such is not required to implement IDE. Such devices may use a Root Complex specific indication that is equivalent to the $\uparrow\downarrow$ XT and \uparrow T $\uparrow\downarrow$ bits \uparrow in the $\uparrow\downarrow$ IDE Prefix (NFM) /OHC-C \downarrow $\uparrow\downarrow$ IDE TLP Prefix (NFM)/ OHC-C \uparrow (FM) to indicate that the TLP is associated with a TVM . For simplicity, such Root Complex specific indications are also referred to as the $\uparrow\downarrow$ XT and \uparrow T $\uparrow\downarrow$ bit, \downarrow $\uparrow\downarrow$ bits, \uparrow although this does not imply an implementation requirement. IDE Stream-specific checks and actions defined in later sections are not required for RCiEPs that do not implement IDE. An RCiEP may not require the use of [Secured SPDM] for protecting the communication between the TSM and DSM if it is possible to ensure the security of communication by other means.

In general, it is not assumed that system configuration relevant to TEE-I/O operation can be protected against inappropriate modification, and in some cases it may not even be possible to do so. Instead, system elements used with TEE-I/O, including TDISP-compliant devices, detect inappropriate modifications when it is possible to do so, and further protect themselves against security policy violations without depending on other elements of the system for assistance, for example by detecting non-IDE TLPs in cases where IDE is required, and by checking memory addresses in received Requests. The detection by one system element of an error condition results in that element entering a “fail safe” error state, but it may not in all cases be possible for this to be directly communicated to other system elements, for example because an attacker may block attempted notifications, and so in such cases error conditions must be inferred, for example through a TDI’s lack of responsiveness.

This chapter specifies the protocol used by the TSM and DSM to associate an IDE ~~↑↓Stream ID~~
~~↑↓Stream ID~~ to be used by the TDI. ~~The T bit allows the originator of a TLP to indicate that a TLP is associated with a TVM. The T bit is used by the device and the host translation agent (TA) to provide access control to TVM assigned memory and memory mapped I/O registers. The T bit must be used only as defined for TDISP.~~ ECN: Base 6.3 XT△↔

The DSM provides the following functions:

1. Authentication of device identities and measurement reporting.
2. Configuring the IDE encryption keys in the device.
3. Device interface management for locking TDI configuration, reporting TDI configurations, attaching, and detaching TDIs from TVMs.
4. Implementing access control and security mechanisms to isolate TVM provided data from entities not in the TCB of the TVM.

The TSM provides the following functions:

1. Provide interfaces to the VMM to assign memory, CPU, and TDI resources to TVMs.
2. Implements the security mechanisms and access controls (e.g., IOMMU translation tables, etc.) to protect confidentiality and integrity of the TVM data and execution state in the host from entities not in the TCB of the TVM.
3. Use TDISP protocol to manage the security state of the TDIs to be used by TVMs.
4. Establishing/managing IDE encryption keys for the host, and, if needed, scheduling key refreshes.

Secured messages as specified in § Section 6.31 are used by TSM and DSM to communicate TDISP messages securely. The secure session establishment is used by the TSM to authenticate the DSM (Optionally, the DSM may be configured to authenticate the TSM, if required by the system design), negotiate cryptographic parameters, and establish shared keying material.

Once the SPDM secure session has been established, the session enters the application phase where all application data between the TSM and DSM are communicated using secured messages within the SPDM secure session. Two types of application data are used by TEE-I/O:

- IDE Key Programming – When IDE is required, the IDE_KM protocol is used for key programming. An IDE stream may also be established between two devices for peer-to-peer communication.
- TDI Management – the TSM uses the TDISP protocol to manage the TDI attach and detach to a TVM. The TSM steps the TDI through the TDISP states as part of the TDI lifecycle management process such as:
 - Locking a TDI configuration for assignment of the TDI to the TVM
 - Making the TDI operational if the TVM approves of the device
 - Detaching a previously assigned TDI from a TVM.

The DSM must track the SPDM session that was used to establish the IDE keys for an IDE stream. For the IDE stream to be usable for carrying TVM data, all the IDE keys for the IDE stream that will be used by the TDI assigned to a TVM must be programmed by the TSM.

Multiple TDIs (e.g., SR-IOV VFs) in a device may generate or receive transactions over the IDE stream established by the TSM and DSM to secure the communication links between the host and the device. One or more of these TDIs may be assigned to TVMs, and one or more of these TDIs may be assigned to legacy VMs. The TSM manages and tracks the TDISP state associated with the TDIs assigned to TVMs.

11.2 TDISP Rules §

Base 6.4 vs Base 6.3

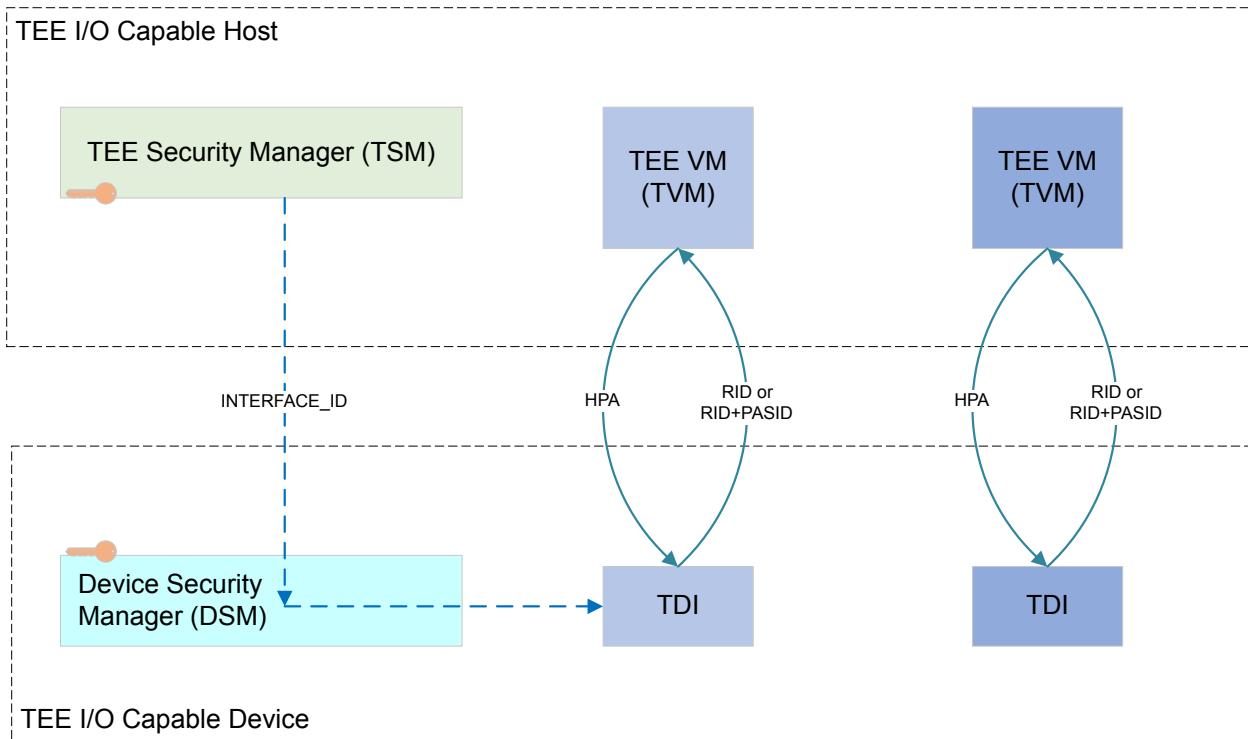


Figure 11-3 Identification of Requests §

As illustrated in § Figure 11-3 , a TDI is managed by a specific DSM , and within the domain of that DSM it is necessary for each TDI to have a unique identifier, called an INTERFACE_ID, that is used in all TSM/DSM messages to indicate the applicable TDI. MMIO requests originated from the TVM are translated by the host and directed to appropriate TDI based on the HPA. Requests generated by the TDI contain the Requester ID (RID) of the function hosting the TDI, and may also have a PASID.

The INTERFACE_ID is composed of a FUNCTION_ID field that identifies the function of the device hosting the TDI and a Reserved field provided for future expansion (see § Figure 11-4 and § Table 11-2). Within the FUNCTION_ID, the Function Number and Device Number are assigned by the device/DSM. The Bus Number and Segment Number are assigned during system enumeration and must not be changed for a TDI in CONFIG_LOCKED and RUN (see below). If the Segment Number is known to the device, and Requester Segment Valid is Set, then the Requester Segment value must match the Segment Number for the device. The DSM must ensure that only valid TDIs are addressed.



Figure 11-4 TDI Identifier – INTERFACE_ID §

Table 11-2 INTERFACE_ID Definition §

| Offset | Field | Size (Bytes) | Description |
|--------|-------------|--------------|---|
| 0 | FUNCTION_ID | 4 | Identifies the function of the device hosting the TDI: 15:0 Requester ID 23:16 Requester Segment (Reserved if Requester Segment Valid is Clear) 24 Requester Segment Valid 31:25 Reserved |
| 4 | Reserved | 8 | Reserved |

Each TDI in the device is associated with a TDISP state machine (see § Figure 11-5).

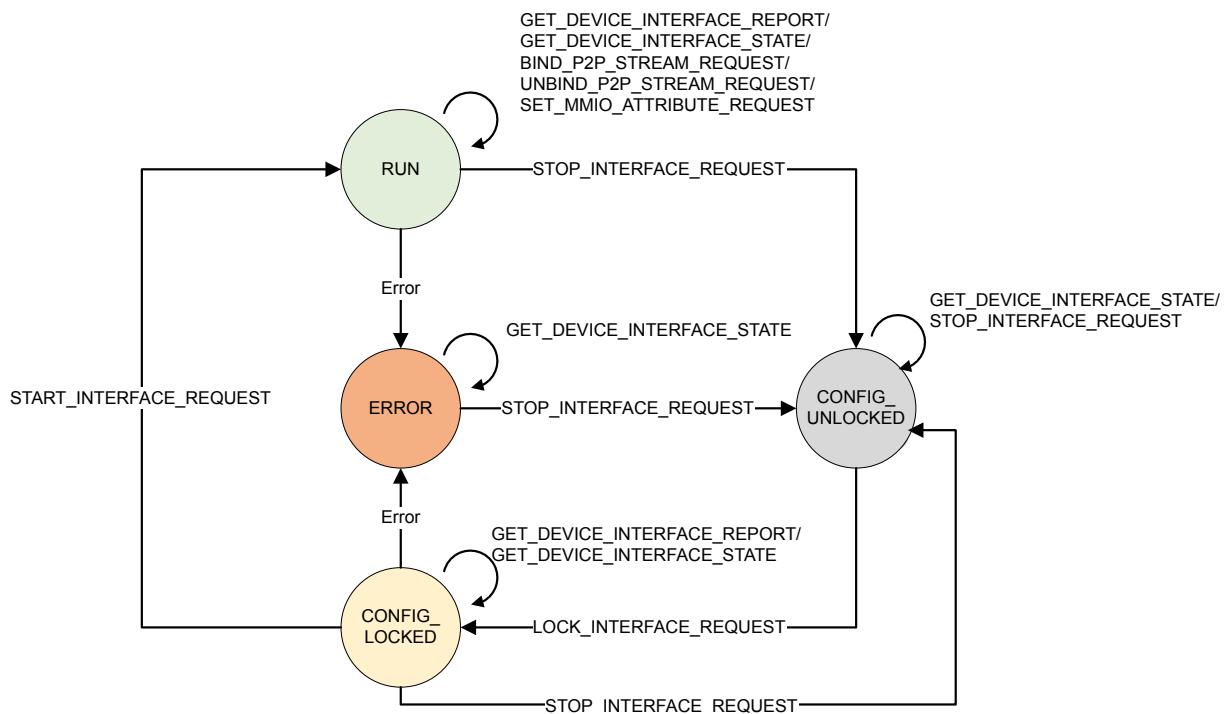


Figure 11-5 TDISP State Machine §

The TSM steps the TDI through these security states as part of the TDI security lifecycle management process, such as locking a TDI configuration in preparation for assignment of the TDI to the TVM, transitioning the TDI to the operational state, and detaching the TDI from a TVM. A TDI is considered “locked” in CONFIG_LOCKED, and RUN. A TDI is considered “unlocked” when in ERROR and CONFIG_UNLOCKED.

For a TDI that supports assignment to Legacy VMs, if a TDI is assigned to a Legacy VM, the VMM assigns the TDI in CONFIG_UNLOCKED, and the TSM must ensure that the TDI remains in that state unless and until the TDI is removed from the Legacy VM and prepared for re-assignment to a TDI.

TDISP requires certain TLPs to be rejected, which means that first the TLP is processed according to the same rules that apply to all other TLPs, and only then are the TDISP-specific rules applied. If the result is a determination that the TLP must be rejected, the associated TDI must transition to ERROR where indicated, but no further error reporting or logging is required to be performed on that TLP, although it is optionally permitted on a case-by-case basis that a Request be handled as an Unsupported Request, and/or a Completion be handled as an Unexpected Completion, or that the TLP be dropped.

IDE Streams that are bound for use with TDISP are permitted to be used for non-TDISP TLPs as well, however:

- for such TLPs the $\uparrow\downarrow\text{XT}$ and $\uparrow\downarrow\text{T}$ $\uparrow\downarrow\text{bit}\downarrow$ $\uparrow\downarrow\text{bits}\uparrow$ must be Clear in Transmitted TLPs, and must be ignored in Received TLPs (as defined in § Section 6.33.4). ECN: Base 6.3 XT Δ
- the DSM must ensure that such TLPs are not allowed to access TVM confidential data.

Security properties for each state and transition rules are as follows:

- **CONFIG_UNLOCKED**

- This is the default state. In CONFIG_UNLOCKED the VMM configures the TDI to be assigned to a TVM.
- A TDI is not required to protect confidential data placed into it in this state; TVMs should not place confidential data into a TDI in this state.
- Memory Requests originating within a TVM (as indicated by the T bit being Set) must be rejected in this state (see § Section 11.2.1)
- This state must be entered from any other state in response to the STOP_INTERFACE_REQUEST message. When the TDI transitions to CONFIG_UNLOCKED, the DSM must ensure all TVM confidential data held by the device in the context of that TDI cannot be exposed as plaintext outside the device and, to the maximum extent possible, the ciphertext associated with TVM data must not be exposed outside the device.

- **CONFIG_LOCKED**

- Once the TDI configuration is finalized by the VMM, the VMM requests the TSM to lock the TDI configuration by transitioning the TDI to CONFIG_LOCKED.
- The DSM must transition a TDI to CONFIG_LOCKED only in response to a LOCK_INTERFACE_REQUEST message.
- Memory Requests originating within a TVM (as indicated by the T bit being Set) must be rejected in this state (see § Section 11.2.1)
- The LOCK_INTERFACE_REQUEST must indicate the $\uparrow\downarrow\text{Stream ID}\downarrow$ $\uparrow\downarrow\text{Stream_ID}\uparrow$ of the IDE stream to bind to the TDI, if IDE is required to secure the transfers to/from the device.
- On entry to this state, the DSM must perform all necessary actions to lock the TDI configuration, and then must start tracking the TDI for changes that affect the configuration or the security of the TDI. Changes detected must be treated as an error, and the TDI transitioned to ERROR. An example list of architectural configurations registers that should be locked and tracked is shown in § Section 11.2.6. It is typically required for the DSM to track additional device-specific configurations, such as the

configuration of work queues, device specific configurations such as MAC address, storage volume, etc.

- The TVM may obtain the identity and measurements of the device hosting the TDI from the DSM , and also, if applicable, verify that an IDE stream has been established by the TSM between the host and the device. The TVM may request the TSM to obtain the TDI configurations using the GET_DEVICE_INTERFACE_REPORT request from the DSM . The TVM may then evaluate the device identity and measurements, in addition to the TDI report to determine if the device meets the security requirements of the TVM .
- If the TVM approves of the device, the TSM may request the TSM to transition the TDI to RUN.

- **RUN**

- TDI resources are operational and permitted to be accessed and managed by the TVM .
- On entry to this state, the DSM must continue tracking the TDI for changes that affect the configuration or the security of the TDI. Changes detected must be treated as an error, and the TDI transitioned to ERROR.

- **ERROR**

- The TDI must not expose confidential TVM data.
- Memory Requests originating within a TVM (as indicated by the T bit being Set) must be rejected in this state (see § Section 11.2.1).
- The TDI must restrict TLP operations as defined in § Section 11.2.1 .
- In ERROR, the TDI may still have confidential TVM data, and it is permitted that clearing this data be deferred until the receipt of a STOP_INTERFACE_REQUEST to transition the TDI to CONFIG_UNLOCKED.
 - It is permitted, but not required, that the TDI transition automatically from ERROR to CONFIG_UNLOCKED, if and only if the TDI first clears all TVM confidential data.

In CONFIG_LOCKED and RUN, the following conditions must be treated as errors, and cause the TDI to move to ERROR:

- Changes to TDI configuration that affect the configuration or the security of the TDI.
 - The Completion Status of Configuration Writes that modify TDI configuration must not be affected.
 - See also § Section 11.2.6 .
- Changes to the Requester ID
- Resetting the TDI using a Function Level Reset. A PF reset affects all subordinate VF **↑↑or SDI↑** TDIs, whereas a VF **↑↑or SDI↑** reset affects only that TDI.
- Any IDE stream bound to the TDI transitions to the Insecure state.
- Except when the TDI implements mechanisms to recover from and/or suitably handle, receipt of a poisoned TLP or detecting data integrity errors in the device for data associated with that TDI, where the error is not recoverable.
- Other device specific conditions or changes in configuration that affect trust properties.

 ECN: Base 6.3
 SIOV△◀▶

The STOP_INTERFACE_REQUEST message may be used to transition the TDI to CONFIG_UNLOCKED. When the TDI transitions back to CONFIG_UNLOCKED, it must ensure that all TVM confidential data held by the device cannot be exposed as plaintext outside the device. To the maximum extent possible the ciphertext associated with TVM confidential data must not be exposed outside the device.

The TSM is permitted to issue a GET_DEVICE_INTERFACE_REPORT in CONFIG_LOCKED and RUN.

The TSM is permitted to issue a GET_DEVICE_INTERFACE_STATE request in all states.

Some TDIs may support peer-to-peer communication with other devices. The ~~Stream ID~~ Stream_ID of the IDE stream(s) used for this communication are configured into the device using the BIND_P2P_STREAM_REQUEST message. This message must only be accepted by the DSM if the TDI is in RUN. The device must be ATS capable, and must have ATS enabled, to support peer-to-peer communication between TVM-assigned TDIs. The TSM and VMM must coordinate the use of ACS mechanisms to redirect device peer-to-peer traffic to the Root Complex, and the TSM must only issue a BIND_P2P_STREAM_REQUEST if the TLPs to be associated with that Selective IDE Stream will, in fact, travel between the two peer devices and not to/from the Root Complex.

Certain configurations require all requests originating from a TDI to be sent to the Root Complex, even if the device is in possession of a Translated Address that appears to refer to a resource within the TDI or elsewhere within the device. Devices are strongly encouraged to implement redirection functionality to send all such requests to the Root Complex. A DSM must advertise the availability of such functionality so that the TSM can establish the correct configuration for the TDI and the rest of the system.

Some TDIs may support updating attributes of one or more MMIO ranges associated with the TDI using the SET_MMIO_ATTRIBUTE_REQUEST.

A TDI must not rely on I/O resources and I/O requests for providing functionality to a TVM. I/O resources must not be able to compromise the confidentiality or integrity of TVM data.

It is not required that Configuration Requests to a TDI be secured.

11.2.1 TDISP TLP Rules §

This section defines the rules for TLPs that are associated with TVMs. In cases where it is possible to ensure communication is not tampered with, it may be possible to avoid the use of IDE, but some equivalent constructs defined by IDE, particularly the T ~~bit~~ bit (and the XT bit),¹ may still be required, and how this is done is outside the scope of this specification.

Under all circumstances, devices must ensure ~~that~~ that

ECN: Base 6.3 XT△↔

ECN: Base 6.3 XT△↔

- ~~The~~ device memory with the IS_NON_TEE_MEM attribute Clear can only be read/written within the context of an authorized TVM (indicated, when IDE is used, by the T bit being ~~Set if the XT Mode is enabled, by the XT bit also being~~ Set). ECN: Base 6.3 XT△↔
- ~~Memory Requests, including ATS Translation Requests, other than MSI/MSI-X interrupts must not be issued by the TDI while in CONFIG_LOCKED or ERROR.~~¹
- ~~TLPs with the XT bit and/or the T bit Set must only be transmitted on an IDE stream bound to the TDI. All TLPs not associated with a peer-to-peer IDE stream must use the IDE stream bound to the TDI by the LOCK_INTERFACE_REQUEST.~~¹

In all cases, traffic that has no security requirement is not required by TDISP to be secured by IDE or other means, although in specific platforms there may be additional security requirements, for example to apply IDE to all TLPs between the host and a particular device.

~~The~~

~~Untranslated Memory Requests must be generated over the IDE stream bound to the TDI by the LOCK_INTERFACE_REQUEST.~~¹ ECN: Base 6.3 XT△↔

~~If the XT Mode is not supported or not enabled, the~~¹ following rules apply to the TDI acting as a Requester: ECN: Base 6.3 XT△↔

- **In systems where IDE is required, a) Untranslated Memory Requests other than MSI/MSI-X interrupts issued by the TDI in CONFIG_LOCKED or RUN must transmit TLPs with T bit set**
 - **For Untranslated Memory Requests issued by the TDI while in RUN, the corresponding Completion(s) must be handled normally if and only if the TDI is still in RUN, and must otherwise be rejected.**
 - All TLPs not associated with a peer to peer IDE Stream must use ignore the Default IDE Stream, if one has been configured. value of the T bit in Received Completions.

↑↑If the XT Mode is enabled, the following rules apply to the TDI acting as a Requester:↑

ECN: Base 6.3 XT△◀▷

- For **Untranslated Memory Reads** issued by the TDI while in RUN, the corresponding Requests must be handled normally if and only if the TDI is still in RUN, and must otherwise be rejected.
 - If the access must be restricted to TEE memory, TDI must Set the XT bit and Set the T bit.
 - If the access must be restricted to non-TEE memory, TDI must Set the XT bit and Clear the T bit.
 - If there are no specific restrictions determined by the TDI for the access, TDI must Clear the XT bit and Set the T bit.
 - For **Untranslated Memory Reads** issued by the TDI while in RUN, the corresponding Requests must be handled normally if and only if the TDI is still in RUN, and must otherwise be rejected.
 - A TDI in RUN must ignore the value of the XT bit and the T bit in Received Completions.

↑↑Untranslated↑ Memory ↑↓Writes other than MSI/MSI-X interrupts must only be↓
↑↑Requests↑ issued ↑↓while↑ ↑by the TDI not↑ in RUN ↑↓and↓ must ↑↓Set↓ ↑Clear the
XT bit and Clear↑ the T bit.

ECN: Base 6.3 XT△◁▷

The MSI capability in the Configuration Space of the function hosting the TDI is not required to have a trusted configuration. With MSI-X, it is possible for the TVM to program the MSI-X table and MSI-X PBA in a trusted manner.

11 of 11

If the XT Mode is not supported or not enabled, it is permitted for a TDI in CONFIG_LOCKED to issue an MSI/MSI-X interrupt only if the T bit is Clear. TDIs in RUN must observe the following rules:

- **↓↓An↓** **↑↑Untranslated Memory Requests (or ATS Translation Requests) associated with↑ MSI ↑↓interrupt↓** **↑↑using address/data pair obtained from the MSI capability of the function↑** must be generated with **↑↑the↑ T bit Clear.** ECN: Base 6.3 XT△
 - **↓↓An↓** **↑↑Untranslated Memory Requests (or ATS Translation Requests) associated with MSI using address/ data pair obtained from the↑ MSI-X ↑↓interrupt↓** **↑↑capability of the TDI↑** must be generated with **↑↑the↑ T bit Clear if the MSI-X table is not part of the MMIO ranges that are locked and reported in the DEVICE_INTERFACE_REPORT, else the↑↑MSI must be generated with the T bit Set.↑**

↑↑If the XT Mode is enabled, it is permitted for a TDI in CONFIG_LOCKED to issue an MSI/MSI-X only if the XT bit is Clear and the T bit is Clear. TDIs in RUN must observe the following rules of generating memory write requests to request service using MSI:↑

ECN: Base 6.3 XT△◀▷

- ↑↑Untranslated Memory Requests (or ATS Translation Requests) associated with MSI using address/data pair obtained from the MSI capability of the function must be generated with the XT bit Clear and the T bit Clear.

- ↑↑↑Untranslated Memory Requests (or ATS Translation Requests) associated with MSI using address/data pair obtained from the↑ MSI-X ↑↓interrupt↑ ↑↑capability of the TDI↑ must be generated with ↑↑↑the XT bit Clear and the T bit Clear if the MSI-X table is not part of the MMIO ranges that are locked and reported in the DEVICE_INTERFACE_REPORT, else the MSI must be generated with the XT bit as XT_BIT_FOR_LOCKED_MSIX and the↑ T bit Set.

If a TEE-I/O capable device supports locking and reporting of MSI-X table, it must allocate the MSI-X table and PBA such that the entirety of the MSI-X table and PBA may be mapped onto separate isolated processor pages. The MSI-X Table and the PBA base addresses must each be aligned to the minimum processor page size on supported platforms. 64KB alignment is recommended to provide compatibility across various processor architectures.

TEE-I/O capable devices must locate the ST table, if supported, in the MSI-X table, if ST mode of operation is supported.

↑↓The↓

↑↑↑If the XT Mode is not supported or not enabled, the↑ following rules apply to the TDI acting as a Completer: ECN: Base 6.3 XT△↔

- Received Memory Requests targeting device memory with the IS_NON_TEE_MEM (see § Section 11.3.22) attribute Clear must be handled normally if and only if all of the following conditions are satisfied:
 - the T bit for the Request is Set
 - the TDI is in RUN
 - if IDE is required, the Request is received on a ↑↓Stream ID↓ ↑↑Stream_ID↑ bound to the TDI;
 in all other cases, the Request must be rejected.
- The TDI's handling is not modified by TDISP state for Received Memory Requests targeting MMIO with IS_NON_TEE_MEM Set.
- The value of the T bit in the Completion(s) returned by the TDI must match the value of the T bit in the corresponding Request.

↑↑↑If the XT Mode is enabled, the following rules apply to the TDI acting as a Completer:↑

ECN: Base 6.3 XT△↔

- ↑↑↑Received Memory Requests targeting device memory with the IS_NON_TEE_MEM (see § Section 11.3.22) attribute Clear must be handled normally if and only if all of the following conditions are satisfied:↑
 - ↑↑↑the XT bit for the Request is Set↑
 - ↑↑↑the T bit for the Request is Set↑
 - ↑↑↑the TDI is in RUN↑
 - ↑↑↑if IDE is required, the Request is received on a Stream ID bound to the TDI; in all other cases, the Request must be rejected.↑
- ↑↑↑Received Memory Requests targeting device memory with the IS_NON_TEE_MEM attribute Set must be handled normally if and only if the XT bit is Set or Clear and the T bit is Clear for the Request; in all other cases, the Request must be rejected.↑
- ↑↑↑The TDI's handling of Received Memory Requests is not modified by the Address Type (AT) field of the Request.↑
- ↑↑↑The value of the XT bit and the T bit in the Completion(s) returned by the TDI must match the value of the XT bit and the T bit in the corresponding Request.↑

There are specific requirements for ↑↓ATS Invalidate↓ ↑↑↑the TLPs associated with the Address Translation Services (ATS) – Translation Requests, ↑↓Invalidate↓ ↑↑Translation↑ Completions, ↑↑↑Translated Requests, Completions for the Translated Requests, Invalidate Request Messages, Invalidate Completion

ECN: Base 6.3 XT△↔

Messages, Page Request Messages and PRG **Responses,** **Response Messages – as** defined in § Section 11.4.10 . VDMs not defined by PCI-SIG must be specified by the Vendor as TDI-specific or not TDI-specific. A TDI is permitted to Transmit and to handle normally Received TDI-Specific Messages while in CONFIG_LOCKED, RUN, and ERROR if and only if the T bit is **Set, if the XT Mode is enabled, the XT bit is also Set.**

11.2.2 TDISP Message Transport §

All TDISP messages must be transported between TSM and DSM using secured messages as specified by Secured CMA/SPDM (see § Section 6.31.4) used within a secure session established between TSM and DSM as specified by [SPDM].

TDISP requires the TSM and the DSM to support AES-256-GCM as the Authenticated Encryption with Associated Data (AEAD) algorithm to protect data transferred using secured messages. The random data field of the secured messages must not be used for TDISP messages or IDE Key Management protocol messages, and this field must have a length of zero.

Function 0 of the device must support the DOE Extended Capability to establish the SPDM session and transport the secured messages.

The [SPDM] Requester role is assumed by the TSM in the host and the Responder role is assumed by the DSM . The TSM is permitted to use an untrusted channel (e.g., proxy through the VMM) to access the transport mechanism.

DOE error conditions only impact TDISP state if the DOE error itself causes an unrecoverable condition, for example by rendering the secure session unusable.

TDISP messages are transported as follows:

- The Requester (TSM) must use the [SPDM] VENDOR_DEFINED_REQUEST format
- The Responder (DSM) must use the [SPDM] VENDOR_DEFINED_RESPONSE format
- The StandardID field of VENDOR_DEFINED_REQUEST and VENDOR_DEFINED_RESPONSE message must contain the value assigned in [SPDM] to identify PCI-SIG.
- The VendorID field of VENDOR_DEFINED_REQUEST and VENDOR_DEFINED_RESPONSE message must contain the value assigned to identify PCI-SIG.
- The first byte of the VendorDefinedReqPayload/VendorDefinedRespPayload is the Protocol ID, and must contain the value 01h to indicate TDISP.
- The TDISP message forms the request or response payload in the VendorDefinedReqPayload or VendorDefinedRespPayload, respectively.
- The VENDOR_DEFINED_REQUEST/VENDOR_DEFINED_RESPONSE must in turn form the Application Data field of a Secured Message per [Secured SPDM].

The encapsulation is as illustrated in § Figure 11-6 :

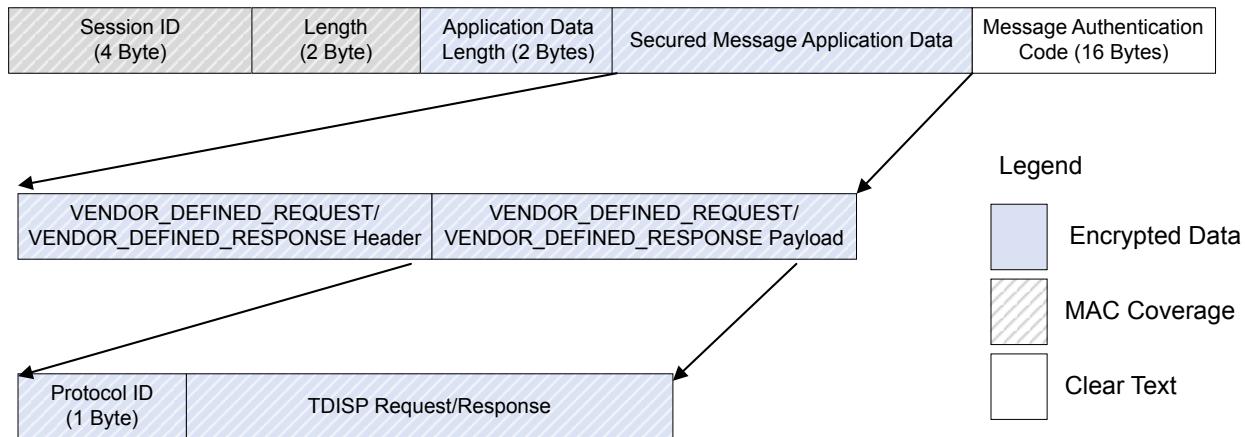


Figure 11-6 TDISP Request/Response Encapsulation §

If a TDISP message is received that has not been transferred securely per [Secured SPDM], the received TDISP message must not be used, and must not result in a response.

11.2.3 Requirements for Requesters (TSM) §

A Requester must not exceed the number of allowed outstanding requests to a specific DSM as indicated by NUM_REQ_ALL, and for a specific TDI as indicated by NUM_REQ_THIS (see § Table 11-10). If the Requester has sent a request to a Responder and wants to send a subsequent request to the same Responder, then the Requester must wait to send the subsequent request until after the Requester completes one of the following actions:

1. Receives the response from the Responder for an outstanding request.
2. Times out waiting for a response.
3. Receives an indication, from the transport layer, that transmission of the request message failed.

A Requester is permitted to send simultaneous request messages to different Responders.

11.2.4 Requirements for Responders (DSM) §

A Responder is not required to process more than NUM_REQ_THIS requests at a time. A Responder that is not ready to accept a new request message must either respond with a TDISP_ERROR response message with ERROR_CODE=BUSY or silently discard the request message.

If a Responder is working on a request message from a Requester, the Responder is permitted to respond with ERROR_CODE=BUSY.

If a Responder enables simultaneous communications with multiple Requesters, the Responder is expected to distinguish the Requesters by using the Session ID in the Secured Message.

11.2.5 TDISP Timing Requirements §

TDISP inherits timing requirements from the SPDM protocol used to encapsulate the messages.

11.2.6 DSM Tracking and Handling of Locked TDI Configurations (Informative) §

The DSM must track attempts to modify registers or other changeable device configuration controls affecting any Physical Function, Virtual Function, or non-IOV Function hosting a TDI in CONFIG_LOCKED or RUN. Precisely what must be tracked is implementation specific, and because the security properties of the device depend on correct implementation of these mechanisms, it is strongly recommended that persons skilled in building and validating secure hardware be deeply involved in the design and validation for all implementations.

§ Table 11-3 provides some general guidance regarding architecturally defined registers. Device-specific registers must be evaluated by the device vendors to determine if modifications to those are allowable. Device vendors must also evaluate additional device specific registers that are mapped into memory space or configuration space of the device to determine if they must be locked and tracked for modifications.

Read-only registers, hardware initialized registers, and registers used as selectors for reading out data (e.g., the Power Budgeting Data Select register) are excluded from this table. The DSM must ensure that attempts to modify those registers cannot affect the security of the TDI.

Table 11-3 Example DSM Tracking and Handling for Architected Registers §

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|--|---|--|
| <u>Cache Line Size</u> , <u>Latency Timer Interrupt Line</u> | Allowed | |
| <u>Command Register</u> | See description | <p>Clearing any of the following bits causes the TDI hosted by the Function to transition to ERROR:</p> <ul style="list-style-type: none"> • <u>Memory Space Enable</u> • <u>Bus Master Enable</u> <p>Modification of other bits is allowed.</p> |
| <u>Status Register</u> | Allowed | |
| <u>BIST Register</u> , <u>Base Address Registers</u> , <u>Expansion ROM Base Address</u> | Error Register | Transition hosted TDI to ERROR. |
| <u>PCI Power Management Capability</u> | See description | If a power transition leads to the function losing its state, then the device transitions the TDI hosted by that function to ERROR. |
| <u>Device Control Register</u> , <u>Device Control 2 Register</u> , <u>Device Control 3 Register</u> | See description | <p>Modifying state of any of the following bits causes the TDI hosted by the function to transition to ERROR:</p> <ul style="list-style-type: none"> • <u>Extended Tag Field Enable</u> • <u>Phantom Functions Enable</u> |

Base 6.4 vs Base 6.3

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|--|---|--|
| | | <ul style="list-style-type: none"> • <u>Initiate Function Level Reset</u> • <u>Enable No Snoop</u> • <u>10-bit Tag Requester Enable</u> • <u>14-bit Tag Requester Enable</u> <p>Modification of other bits is allowed.</p> |
| <u>Device Status Register , Device Status 2 Register , Device Status 3 Register</u> | Allowed | |
| <u>Link Control Register , Link Control 2 Register , Link Control 3 Register , 16.0 GT/s Control Register , 32.0 GT/s Control Register , 64.0 GT/s Control Register</u> | Allowed | If modifications to these registers lead to a Link Down condition, the IDE streams configured in the device transition to the Insecure state, and as a result TDIs bound to those streams transition to ERROR. |
| <u>Link Status Register , Link Status 2 Register , 16.0 GT/s Status Register , 32.0 GT/s Status Register , 64.0 GT/s Status Register</u> | Allowed | |
| <u>MSI Capability</u> | Allowed | |
| <u>MSI-X Capability</u> | See description | If MSI-X table was locked and reported, then any modifications cause transition to ERROR. Modifications are allowed otherwise. |
| <u>Secondary PCIe Extended Capability , Physical Layer 16.0 GT/s Extended Capability , Physical Layer 32.0 GT/s Extended Capability , Physical Layer 64.0 GT/s Extended Capability , Lane Margining at the Receiver Extended Capability , Flit Error Injection Extended Capability</u> | Allowed | If modifications to these registers lead to a Link Down condition, the IDE streams configured in the device transition to the Insecure state, and as a result TDIs bound to those streams transition to ERROR. |
| <u>ACS Extended Capability , Latency Tolerance Reporting Extended Capability</u> | Allowed | |
| <u>L1 PM Substates Extended Capability</u> | Allowed | If modifications to this register lead to a Link Down condition, the IDE streams configured in the device transition to the Insecure state, and as a result TDIs bound to those streams transition to ERROR. |
| <u>Advanced Error Reporting Extended Capability</u> | Allowed | As specified in § <u>Section 6.2.3.2.2</u> , error mask register settings control reporting of detected errors, but do not block error detection. |
| <u>Enhanced Allocation Capability , Resizable BAR Extended Capability , VF Resizable BAR Extended Capability , ARI Extended Capability , PASID Extended Capability</u> | Error | Transition the TDI to ERROR. |

Base 6.4 vs Base 6.3

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|--|---|--|
| <u>Virtual Channel Extended Capability , Multi-Function Virtual Channel Extended Capability</u> | Allowed – see description | Device must enforce transaction ordering when TC/VC mapping is changed, or arbitration tables are updated. |
| <u>Vendor Specific Capability , Vendor Specific Extended Capability , Designated Vendor Specific Extended Capability</u> | See description | To be analyzed by the vendor based on the security principles provided by TDISP. |
| <u>Multicast Extended Capability</u> | Error | Enabling multicast mechanism is not supported for TEE-I/O capable devices. Transition hosted TDIs to ERROR. |
| <u>Dynamic Power Allocation Extended Capability</u> | Allowed | A device must guard against the part being placed outside of its specification. If the device cannot reliably operate within the power allocation through mechanisms like throttling, frequency control, etc. then the device transitions hosted TDIs to ERROR. |
| <u>TPH Requester Extended Capability</u> | See description | To be analyzed by the vendor based on the security principles provided by TDISP. |
| <u>Precision Time Measurement Extended Capability , Hierarchy ID Extended Capability , Native PCIe Enclosure Management Extended Capability , Alternate Protocol Extended Capability</u> | Allowed | |
| <u>Protocol Multiplexing Extended Capability</u> | Allowed | A TEE-I/O capable device that supports PMUX must not transmit or receive transactions for TDIs in CONFIG_LOCKED or RUN using PMUX packets. If modifications to this register lead to a Link Down condition, the IDE streams configured in the device transition to the Insecure state, and as a result TDIs bound to those streams transition to ERROR. |
| <u>Shadow Functions Extended Capability</u> | Not Applicable | Use of ↓↓shadow functions↓↓ is not permitted for TEE-I/O usages. |
| <u>Data Object Exchange Extended Capability</u> | Allowed | |
| <u>Integrity and Data Encryption Extended Capability</u> | See description | Modifying the stream control register, selective IDE RID association registers, or selective IDE address association registers of streams that are bound to locked TDIs is an error, and the device transitions the TDIs bound to such stream to ERROR. |
| <u>ATS Extended Capability</u> | ↓↓Allowed↓↓ ↑↑Error↑↑ | ECN: Base 6.3 XT△↔ ↓↑Transition the TDI to ERROR.↑↑ |
| <u>Page Request Extended Capability , SR-IOV Extended Capability</u> | Error | Transition hosted TDIs to ERROR. |

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|---|---|-------------|
| VPD Capability | Allowed | |

11.2.7 TVM Acceptance of a TDI §

A TVM must ask the following questions before it accepts a TDI into its TCB:

1. Is the identity of the device and the measurements reported by the device hosting the TDI acceptable?
2. Is there ~~the~~ an SPDM secure session established between the TSM and the DSM, and does the identity authenticated by the TSM to setup the SPDM secure session match the identity reported to the TVM?
3. When IDE is required, were all IDE keys for the IDE stream used by the TDI established or verified by the TSM?
4. Has the VMM configured the TDI and mapped the TDI into the TVM address space as expected?

The TVM queries the TSM using a TSM-provided interface to determine the answers to questions 1, 2 and 3. The TVM then requests a report of the TDI configuration (see § Section 11.3.10) from the DSM and may use it, with support from the TSM to determine the answer to question 4.

If the answer to all of these questions is a yes, then the TVM may accept the TDI into its TCB.

11.3 TDISP Message Formats and processing §

11.3.1 TDISP Request Codes §

§ Table 11-4 defines the TDISP request codes. All TDISP-compatible implementations must use the following TDISP request codes. Unsupported request codes must return a TDISP_ERROR response message with ERROR_CODE=UNSUPPORTED_REQUEST.

Table 11-4 TDISP Request Codes §

| Message | Code Value | Required/Optional for Device | Legal TDISP states | Description |
|-----------------------------|------------|------------------------------|--|---|
| GET_TDISP_VERSION | 81h | Required | N/A | This request message must retrieve a device's TDISP version |
| GET_TDISP_CAPABILITIES | 82h | Required | N/A | Retrieve protocol capabilities of the device |
| LOCK_INTERFACE_REQUEST | 83h | Required | CONFIG_UNLOCKED | Move TDI to CONFIG_LOCKED |
| GET_DEVICE_INTERFACE_REPORT | 84h | Required | CONFIG_LOCKED, RUN | Obtain a TDI report |
| GET_DEVICE_INTERFACE_STATE | 85h | Required | CONFIG_UNLOCKED, CONFIG_LOCKED, RUN, ERROR | Obtain state of a TDI |
| START_INTERFACE_REQUEST | 86h | Required | CONFIG_LOCKED | Start a TDI |

| Message | Code Value | Required/ Optional for Device | Legal TDISP states | Description |
|-----------------------------|--------------------|----------------------------------|--|---|
| STOP_INTERFACE_REQUEST | 87h | Required | CONFIG_UNLOCKED, CONFIG_LOCKED, RUN, ERROR | Stop and move TDI to CONFIG_UNLOCKED (if not already in CONFIG_UNLOCKED) |
| BIND_P2P_STREAM_REQUEST | 88h | Optional | RUN | Bind a P2P stream |
| UNBIND_P2P_STREAM_REQUEST | 89h | Optional | RUN | Unbind a P2P stream |
| SET_MMIO_ATTRIBUTE_REQUEST | 8Ah | Optional | RUN | Update attributes of specified MMIO range |
| VDM_REQUEST | 8Bh | Optional | N/A | Vendor defined message request |
| ↑↑SET_TDISP_CONFIG REQUEST↑ | ECN: Base 6.3 XT△↔ | ↑↑8Ch↑ | ↑↑Optional↑ | ↑↑CONFIG_UNLOCKED↑ |
| | | | | ↑↑Set TDISP configuration↑ |

11.3.2 TDISP Response Codes §

The Request Response Code field in the response message must specify the appropriate response code (see § Table 11-5) for a request. On a successful completion of an operation, the specified response message must be returned. Upon an unsuccessful completion of an operation, the TDISP_ERROR response message must be returned. Undefined/unsupported response codes must be treated as if they were TDISP_ERROR.

Table 11-5 TDISP Response Codes §

| Message | Code Value(h) | Required/ Optional | Description |
|--------------------------|-----------------|-----------------------|---|
| TDISP_VERSION | ↑↑01↓ ↑↑01h↑ | Required | Version supported by device. |
| TDISP_CAPABILITIES | ↑↑02↓ ↑↑02h↑ | Required | Protocol capabilities of the device. |
| LOCK_INTERFACE_RESPONSE | ↑↑03↓ ↑↑03h↑ | Required | Response to LOCK_INTERFACE_REQUEST |
| DEVICE_INTERFACE_REPORT | ↑↑04↓ ↑↑04h↑ | Required | Report for a TDI |
| DEVICE_INTERFACE_STATE | ↑↑05↓ ↑↑05h↑ | Required | Returns TDI state |
| START_INTERFACE_RESPONSE | ↑↑06↓ ↑↑06h↑ | Required | Response to request to move TDI to RUN |
| STOP_INTERFACE_RESPONSE | ↑↑07↓ ↑↑07h↑ | Required | Response to a STOP_INTERFACE_REQUEST |
| BIND_P2P_STREAM_RESPONSE | ↑↑08↓ ↑↑08h↑ | Optional | Response to bind P2P stream request |

| Message | Code Value(h) | Required/Optional | Description |
|------------------------------|----------------------|-------------------|---|
| UNBIND_P2P_STREAM_RESPONSE | ↑↓09↑ ↑↓09h↑ | Optional | Response to unbind P2P stream request |
| SET_MMIO_ATTRIBUTE_RESPONSE | ↑↓0A↑ ↑↓0Ah↑ | Optional | Response to update MMIO range attributes |
| VDM_RESPONSE | ↑↓0B↑ ↑↓0Bh↑ | Optional | Vendor defined message response |
| ↑↓SET_TDISP_CONFIG_RESPONSE↑ | ECN: Base 6.3 XT△< > | ↑↓0Ch↑ | ↑↓Optional↑ ↑↓Response to a TDISP configuration request↑ |
| TDISP_ERROR | ↑↓7F↑ ↑↓7Fh↑ | Required | Error in handling a request |

Base 6.4 vs Base 6.3

11.3.3 TDISP Message Format and Protocol Versioning §

§ Table 11-6 defines the fields that are included in all TDISP messages. Unless otherwise specified, the following rules shall apply to all request and response messages in TDISP:

- Reserved, unspecified, or unassigned values in enumerations or other numeric ranges are reserved for future definition of the TDISP specification. Reserved numeric and bit fields must be written as zero (0) and ignored when read.
- Byte ordering of multi-byte numeric fields or multi-byte bit fields is "Little Endian" (that is, the lowest byte offset holds the least significant byte, and higher offsets hold the more significant bytes).
- All message fields, regardless of size or endianness, map the highest numeric bits to the highest numerically assigned byte in monotonically decreasing order until the least numerically assigned byte of that field.

All TDISP messages include the one Byte TDISPVersion field, which is divided into two sub-fields

- Bits 7:4 - Major Version – The major version of the TDISP Specification. A device must not communicate by using an incompatible TDISP version value. The Major version is incremented when the protocol modification breaks backward compatibility.
- Bits 3:0 - Minor Version – The minor version of the TDISP specification. A specification with a given minor version extends a specification with a lower minor version if they share the major version. The Minor Version is incremented when the protocol modification maintains backward compatibility.

This version of TDISP is V1.0, represented as 10h.

See § Section 11.3.5 for details on compatible version negotiation.

Table 11-6 TDISP Message Format §

| Offset | Field | Size (Bytes) | Description |
|--------|--------------|--------------|---|
| 0 | TDISPVersion | 1 | The TDISPVersion field represents the version of the specification encoded as follows: <ul style="list-style-type: none"> • Bits 7:4 – Major Version • Bits 3:0 – Minor Version |

| Offset | Field | Size (Bytes) | Description |
|--------|-----------------------|--------------|--|
| 1 | MessageType | 1 | The code identifying the type of the message (see § Table 11-4 and § Table 11-5). |
| 2 | Reserved | 2 | Reserved |
| 4 | INTERFACE_ID | 12 | The TDI's ID, as defined in § Section 11.2 |
| 16 | TDISP message payload | Variable | Zero or more bytes that are specific to the MessageType |

The device must fail the request and return the indicated response code if any of the error cases defined in § Table 11-7 are detected:

Table 11-7 Generic Error Response Codes §

| Error Code | Description |
|-----------------------|---|
| INVALID_REQUEST | One or more of the fields of the request being invalid |
| VERSION_MISMATCH | Protocol version is unsupported or is not the agreed upon version |
| BUSY | Device cannot process the request due to being busy |
| UNSPECIFIED | Error due to an unspecified reason |
| VENDOR_SPECIFIC_ERROR | Error due to a vendor specific reason |
| INVALID_INTERFACE | The INTERFACE_ID indicated is not within the domain of the DSM |

11.3.4 GET_TDISP_VERSION §

This request message must retrieve the device's TDISP version. In all future TDISP versions, the TDISP_GET_VERSION and TDISP_VERSION response messages will be backward compatible with all previous versions. The Requester must begin the discovery process by sending a TDISP_GET_VERSION request message with major version 1h. All Responders must always support TDISP_GET_VERSION request message with major version 1h and provide a TDISP_VERSION response containing all supported versions, as the TDISP_GET_VERSION request message table describes. The Requester must consult the TDISP_VERSION response to select a common (typically highest) version supported. The Requester must use the selected version in all future communication of other requests. A Requester must not issue other requests until it has received a successful TDISP_VERSION response and has identified a common version supported by both sides. A Responder must not respond to TDISP_GET_VERSION request message with ERROR_CODE=RESPONSE_NOT_READY.

A host system is permitted to use GET_TDISP_VERSION to determine if a device ↑↓support↓ ↑↑supports↑ TDISP.

11.3.5 TDISP_VERSION §

Table 11-8 TDISP_VERSION §

| Offset | Field | Size (Bytes) | Description |
|--|-----------------------|--------------|-----------------------------------|
| Payload (All fields in little endian format) | | | |
| 16 | VERSION_NUM_COUNT (N) | 1 | Number of version number entries. |

| Offset | Field | Size (Bytes) | Description |
|--------|------------------------|--------------|---|
| | | | Minimum permitted value is 1. |
| 17 | VERSION_NUM_ENTRY[1-N] | 1 x N | <p>8-bit version entry where each entry is formatted as:</p> <ul style="list-style-type: none"> • 7:4 – Major Version Number • 3:0 – Minor Version Number |

11.3.6 GET_TDISP_CAPABILITIES §

Used to retrieve the Responder's TDISP capabilities. TDISP protocol inherits the SPDM timing specifications, including the timing parameter CT used to determine the time the Responder must respond to a message needing cryptographic processing.

Table 11-9 GET_TDISP_CAPABILITIES §

| Offset | Field | Size(Bytes) | Description |
|--|----------|-------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | TSM_CAPS | 4 | <p>TSM Capability Flags</p> <p>Bits 31:0 – Reserved</p> |

11.3.7 TDISP_CAPABILITIES §

Capabilities supported by Responder.

Table 11-10 TDISP_CAPABILITIES §

| Offset | Field | Size (Bytes) | Description |
|--|--------------------------------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | DSM_CAPS | 4 | <p>DSM Capability Flags</p> <p>Bit 0 XT_MODE_SUPPORTED indicates if the device supports the XT Mode operation or not. ↑↑↑</p> <p>ECN: Base 6.3 XT △↔</p> <p>Bits ↑↑31:0↓ ↑↑31:1↑ – Reserved</p> |
| 20 | REQ_MSGS_SUPPORTED | 16 | Bitmask indicating each type of request message supported by the device, where the bit index corresponds to the message code defined in § Table 11-4 minus 80h. |
| 36 | LOCK_INTERFACE_FLAGS_SUPPORTED | 2 | <p>Bitmask indicating lock interface flags supported by the device, where the bit index corresponds to the FLAGS field definition in § Table 11-11.</p> <p>Lack of support for a specific flag indicates that software must not assume any particular device behavior regarding the related capability, unless it has device-specific knowledge through other means.</p> |

| Offset | Field | Size (Bytes) | Description |
|--------|----------------|--------------|--|
| 38 | - | 3 | Reserved |
| 41 | DEV_ADDR_WIDTH | 1 | Device reports the number of address bits it supports. For example, a value of 52 in this field indicates support for Bits 51:0. |
| 42 | NUM_REQ_THIS | 1 | Number of outstanding Requests permitted by the DSM for this TDI. |
| 43 | NUM_REQ_ALL | 1 | Number of outstanding Requests permitted by the DSM for all TDIs managed by this DSM . |

11.3.8 LOCK_INTERFACE_REQUEST §

The LOCK_INTERFACE_REQUEST is used to move the TDI to CONFIG_LOCKED, provided that the DSM confirms that the device, including elements of Function 0 and the TDI itself, is acceptably configured and in an acceptable state.

The device must fail the request if any of the following errors are detected:

- Interface ID specified in the request is not hosted by the device.
- TDI is not in CONFIG_UNLOCKED.
- For TDIs where IDE is required:
 - The default Stream does not match the **↓↑Stream ID↓↑↑Stream_ID↑** indicated
 - The default stream does not have IDE keys programmed for all sub-streams
 - All IDE keys of the default stream were not configured over the SPDM session on which the LOCK_INTERFACE_REQUEST was received
 - Multiple IDE configuration registers in the device have been configured as the default stream
 - The default Stream is associated with a TC other than TC0
 - **↓↑If the XT Mode is supported by the device, the XT Enable setting for the default Stream is not matching the XT Mode setting configured for the device by SET_TDISP_CONFIG_REQUEST.↑** ECN: Base 6.3 XT△◀
- Phantom Functions Enabled
- Device PF BARs configured with overlapping addresses
- Expansion ROM base address, if supported, configured to overlap with a BAR
- Resizable BAR control registers programmed with an unsupported BAR size
- VF BARs are configured with address overlapping another VF BAR, a PF BAR or Expansion ROM BAR
- **↓↑Ranges of a PF's BAR that are in use by a TDI which is itself a Scalable Device Interface (SDI) overlap ranges of that PF's BARs that are used for any other purpose.↑** ECN: Base 6.3 SIOV△◀
- Unsupported system page size is configured in the system page size register of SR-IOV capability
- Cache Line Size configured for LN requester capability (deprecated in PCIe Revision 6.0), if supported and enabled, does not match the system cache line size specified in the LOCK_INTERFACE_REQUEST or is configured to a value not supported by the device.
- ST mode selected in TPH Requester Extended Capability, if supported and enabled, does not correspond to a mode supported by the function hosting the TDI.

- Other device determined errors in the device or TDI configurations

The LOCK_INTERFACE_REQUEST binds and configures the following parameters into the TDI:

- MMIO_REPORTING_OFFSET – To avoid leaking physical addresses to a TVM, the TSM specifies a MMIO_REPORTING_OFFSET to be applied to all future DEVICE_INTERFACE_REPORT generated by this TDI. MMIO_REPORTING_OFFSET is a signed (2's complement) 64-bit value that must be added to all MMIO physical addresses reported by the TDI. In order to maintain host-specific page alignment, the TSM is permitted to supply the corresponding lower bits of MMIO_REPORTING_OFFSET as zero. The TSM must supply an offset that does not result in overflow/underflow.
- NO_FW_UPDATE when Set indicates that when this TDI is in CONFIG_LOCKED or RUN, the device must not accept firmware updates. This option allows certain TVM to opt-out of further firmware updates to the device once the TVM starts using the TDI. To perform a firmware update, a VMM must detach from the TEEs all TDI that are locked with NO_FW_UPDATE=1 from the TVM and move them to CONFIG_UNLOCKED.
- System cache line size.
- Whether MSI-X table and PBA in the function hosting the TDI must be locked. A TDI is permitted to lock the MSI-X table and PBA even if not directed to do so. If the MSI-X table and PBA are not locked, then they must not be reported in the DEVICE_INTERFACE_REPORT. If the MSI-X table and PBA are locked, transactions to access the MSI-X table and PBA without the T bit Set must be rejected.
- BIND_P2P when Set indicates whether the Direct-P2P (device/device peer-to-peer) support may be enabled later via BIND_P2P_STREAM_REQUEST messages and a valid P2P address mask is specified in the request.
- ALL_REQUEST_REDIRECT when Set indicates that TDI must redirect all ATS Translated Requests upstream to the Root Complex, using the default Selective IDE Stream if one is configured, to perform access checks. This includes TDI accesses to the local resources within TDI or other resources of a device that are based on a Translated Address.

On successful processing of the request, the device responds with a LOCK_INTERFACE_RESPONSE message.

Table 11-11 LOCK_INTERFACE_REQUEST §

| Offset | Field | Size (Bytes) | Description |
|--|---------------------------------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | FLAGS | 2 | <p>Bit 0 NO_FW_UPDATE – When 1, indicates that device firmware updates are not permitted while in CONFIG_LOCKED or RUN. When 0, indicates that firmware updates are allowed while in these states.</p> <p>Bit 1 System Cache Line Size – when 0 indicates the system CLS is 64 bytes and when Set indicates system CLS is 128 bytes.</p> <p>Bit 2 LOCK_MSIX – Lock MSI-X table and PBA.</p> <p>Bit 3 BIND_P2P – When 1, indicates that Direct-P2P support may be enabled later via BIND_P2P_STREAM_REQUEST messages and a valid P2P address mask is specified in this request. When 0, indicates that Direct-P2P is not allowed or enabled for this TDI.</p> <p>Bit 4 ALL_REQUEST_REDIRECT – The TDI must redirect all ATS Translated Requests upstream to the Root Complex to perform access checks.</p> <p>Bits 15:5 Reserved</p> |
| 18 | Stream ID for Default Stream | 1 | Indicates the Stream ID for the Stream configured as the IDE default stream. |

| Offset | Field | Size (Bytes) | Description |
|--------|-----------------------|--------------|--|
| 19 | - | 1 | Reserved |
| 20 | MMIO_REPORTING_OFFSET | 8 | MMIO ranges reported in all DEVICE_INTERFACE_REPORT is reported with this offset added to the physical address |
| 28 | BIND_P2P_ADDRESS_MASK | 8 | Mask to be applied to target addresses for peer-to-peer transaction issued by the TDI using the BIND_P2P_STREAM_REQUEST stream (to clear-out any metadata information embedded in the address). This mask must be applied prior to using the Selective IDE Address Association mechanism. This mask is not applicable or applied for Requests bound to the Root Complex. |

11.3.9 LOCK_INTERFACE_RESPONSE §

LOCK_INTERFACE_RESPONSE is provided on successful handling of the LOCK_INTERFACE_REQUEST and the device having moved the TDI to CONFIG_LOCKED. The response message also provides a START_INTERFACE_NONCE that is generated when the TDI is locked. This nonce should be generated by the device in response to moving the TDI to CONFIG_LOCKED. This nonce must be destroyed when the TDI moves to CONFIG_UNLOCKED or ERROR from CONFIG_LOCKED. See § Section 11.3.14 for additional rules regarding this nonce.

Generating a LOCK_INTERFACE_RESPONSE implies that the device has successfully completed the following operations:

- All in-flight and accepted work for the TDI, before lock request was received, are aborted
- All DMA for the TDI, before the lock request was received, are completed, or aborted
- If function hosting the TDI is capable of Address Translation Service (ATS), all ATS requests for the TDI, generated before the lock request was received, have completed, or aborted. The device must invalidate translations cached in the ATC by the Requester ID of the function hosting the TDI.
- If function hosting the TDI is capable of Page Request Interface Service (PRI), page requests for the TDI, generated before the lock request was received, have received responses or the TDI will discard page responses for outstanding page requests.
- Additional private resources that need to be assigned to the TDI by the DSM at the time of locking the TDI have been successfully allocated and assigned.
- DSM has carried out necessary actions on the device side to lock the TDI configuration and IDE configuration registers for the default stream. DSM has enabled mechanisms to track changes to the configurations of the TDI and the IDE configuration register for the default stream.

It is permitted for a TDI to return INVALID_DEVICE_CONFIGURATION in response to a LOCK_INTERFACE_REQUEST for $\text{implementation-specific}$ reasons.

Table 11-12 LOCK_INTERFACE_RESPONSE §

| Offset | Field | Size (Bytes) | Description |
|--|-----------------------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | START_INTERFACE_NONCE | 32 | Device generated nonce to include in START_INTERFACE_REQUEST message. |

§ Table 11-13 defines the error codes and conditions:

Table 11-13 LOCK_INTERFACE_REQUEST Error Codes §

| Error Code | Description |
|------------------------------|---|
| INVALID_REQUEST | Device supports IDE capability and <ul style="list-style-type: none"> • Keys have not been configured for all sub-streams of the default stream • Keys for the default stream were not configured using the SPDM session on which the LOCK_INTERFACE_REQUEST was received |
| INSUFFICIENT_ENTROPY | The device fails to generate nonce. |
| INVALID_INTERFACE_STATE | If the TDI is not in CONFIG_UNLOCKED. |
| INVALID_DEVICE_CONFIGURATION | Locking the TDI failed due to invalid/unsupported device configurations. |

11.3.10 GET_DEVICE_INTERFACE_REPORT §

The GET_DEVICE_INTERFACE_REPORT is used to request a DEVICE_INTERFACE_REPORT from the device. The DEVICE_INTERFACE_REPORT may, in some cases, be larger than the requester can consume in a single response, so the requester is provided with the means to request a specific portion of the overall DEVICE_INTERFACE_REPORT to be sent with a given response.

The device must fail the request if any of the following errors are detected:

- Interface ID in the request is not hosted by the device
- TDI is not in CONFIG_LOCKED or RUN
- Invalid offset specified

Table 11-14 GET_DEVICE_INTERFACE_REPORT §

| Offset | Field | Size (Bytes) | Description |
|--|--------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | OFFSET | 2 | <p>Offset in bytes from the start of the report to where the read request message begins. The responder must send its report starting from this offset.</p> <p>For first GET_DEVICE_INTERFACE_REPORT request, the Requester must set this field to 0.</p> <p>For non-first requests, Offset is the sum of PORTION_LENGTH values in all previous DEVICE_INTERFACE_REPORT responses.</p> |
| 18 | LENGTH | 2 | <p>Length of report, in bytes, to be returned in the corresponding response. Length is an unsigned 16-bit integer.</p> <p>This value is the smaller of the following values:</p> <ul style="list-style-type: none"> • Capacity of requester's internal buffer for receiving Responder's report. • The REMAINDER_LENGTH of the preceding DEVICE_INTERFACE_REPORT response. <p>For the first GET_DEVICE_INTERFACE_REPORT request, the requester must use the capacity of the requester's receiving buffer. If offset=0 and length=FFFFh, the requester is requesting the entire report.</p> |

| Offset | Field | Size (Bytes) | Description |
|--------|-------|--------------|---|
| | | | The Responder is permitted to provide less than the requested length if the Responder's buffer length is limited. |

11.3.11 DEVICE_INTERFACE_REPORT §

Table 11-15 DEVICE_INTERFACE_REPORT §

| Offset | Field | Size (Bytes) | Description |
|--|------------------|----------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | PORTION_LENGTH | 2 | Number of bytes of this portion of TDI report. This must be less than or equal to LENGTH received as part of the request. For example, the Responder is permitted to set this field to a value less than LENGTH received as part of the request due limitations on the Responder's internal buffer. |
| 18 | REMAINDER_LENGTH | 2 | Number of bytes of the TDI report that have not been sent yet after the current response. For the last response, this field must be 0 as an indication to the Requester that the entire TDI report has been sent. |
| 20 | REPORT_BYTES | PORTION_LENGTH | Requested contents of TDI report |

The TDI report is structured as follows:

Table 11-16 TDI Report Structure §

| Offset | Field | Size (Bytes) | Description |
|--------|-----------------------|--------------|--|
| 0 | INTERFACE_INFO | 2 | <p>Bit 0 When 1, indicates that device firmware updates are not permitted while in CONFIG_LOCKED or RUN. When 0, indicates that firmware updates are permitted while in these states</p> <p>Bit 1 TDI generates DMA requests without PASID</p> <p>Bit 2 TDI generates DMA requests with PASID</p> <p>Bit 3 ATS supported and enabled for the TDI</p> <p>Bit 4 PRS supported and enabled for the TDI</p> <p>Bit ↑XT Mode supported and enabled for the device↑</p> <p>ECN: Base 6.3 XT△<△ ↑Bit 15:5↓ ↑Bit 15:6↓ Reserved</p> <p>↑Bit 5↑</p> |
| 2 | - | 2 | Reserved for future use. |
| 4 | MSI_X_MESSAGE_CONTROL | 2 | MSI-X capability message control register state. Must be Clear if a) capability is not supported or b) MSI-X table is not locked. |
| 6 | LNR_CONTROL | 2 | LNR control register from LN Requester Extended Capability. Must be Clear if LNR capability is not supported. LN is deprecated in PCIe Revision 6.0. |
| 8 | TPH_CONTROL | 4 | TPH Requester Control Register from the TPH Requester Extended Capability. Must be Clear if a) TPH capability is not support or b) MSI-X table is not locked. |

Base 6.4 vs Base 6.3

| Offset | Field | Size (Bytes) | Description |
|-----------------------|------------------------------|--------------|---|
| 12 | MMIO_RANGE_COUNT (N) | 4 | Number of MMIO Ranges in report |
| | | | Each MMIO Range of the TDI is reported with the MMIO reporting offset added. Base and size in units of 4K pages <ul style="list-style-type: none"> • 8 bytes – First 4K page with offset added • 4 bytes - Number of 4K pages in this range • 4 bytes – Range Attributes <ul style="list-style-type: none"> Bit 0 Bit 0 – MSI-X Table – if the range maps MSI-X table. This must be reported only if locked by the LOCK_INTERFACE_REQUEST. Bit 1 MSI-X PBA – if the range maps MSI-X PBA. This must be reported only if locked by the LOCK_INTERFACE_REQUEST. Bit 2 IS_NON_TEE_MEM – must be 1b if the range is non-TEE memory. For attribute updatable ranges (see below), this field must indicate attribute of the range when the TDI was locked. Bit 3 IS_MEM_ATTR_UPDATABLE – must be 1b if the attributes of this range is updatable using SET_MMIO_ATTRIBUTE_REQUEST. Bit 15:4 Reserved Bit 31:16 Range ID – a device specific identifier for the specified range. The range ID may be used to logically group one or more MMIO ranges into a larger range. |
| 16 + N * 16 | DEVICE_SPECIFIC_INFO_LEN (L) | 4 | Number of bytes of device specific information |
| 16 + N * 16 + 4 | DEVICE_SPECIFIC_INFO | L | Device specific information |

A TDI may generate (a) all DMA requests without PASID, (b) all DMA requests with PASID, or (c) some DMA requests with and others without PASID.

The Range ID is used to logically group the ranges reported in the report into logical groups.

MMIO ranges assigned via BAR(s) must be reported in ascending order starting with the lowest numbered BAR such that the first range corresponds to the first BAR and so on. The range ID reports the BAR equivalent Indicator (BEI). Values 0-7 of the Range ID are reserved to indicate the BEI. The device must report the BAR equivalent Indicator (BEI) for ranges associated with a PCIe BAR.

When reporting the MMIO range for a TDI, the MMIO ranges must be reported in the logical order in which the TDI MMIO range is configured such that the first range reported corresponds to first range of pages in the TDI and so on.

The device is permitted to include additional device specific information to the TVM in the report. The device specific information may be used to report configurations of the TDI and/or to enumerate capabilities of the TDI. Example of such device specific information include:

- A network device may include receive-side scaling (RSS) related information such as the RSS hash and mappings to the virtual station interface (VSI) queues, etc.

- A NVMe device may include information about the associated name spaces, mapping of name space to command queue-pair mappings, etc.
- Accelerators may report capabilities such as algorithms supported, queue depths, etc.

The following sequence diagram shows the high-level request-response message flow for Responder response when it cannot return the entire data requested by the Requester in the first response.

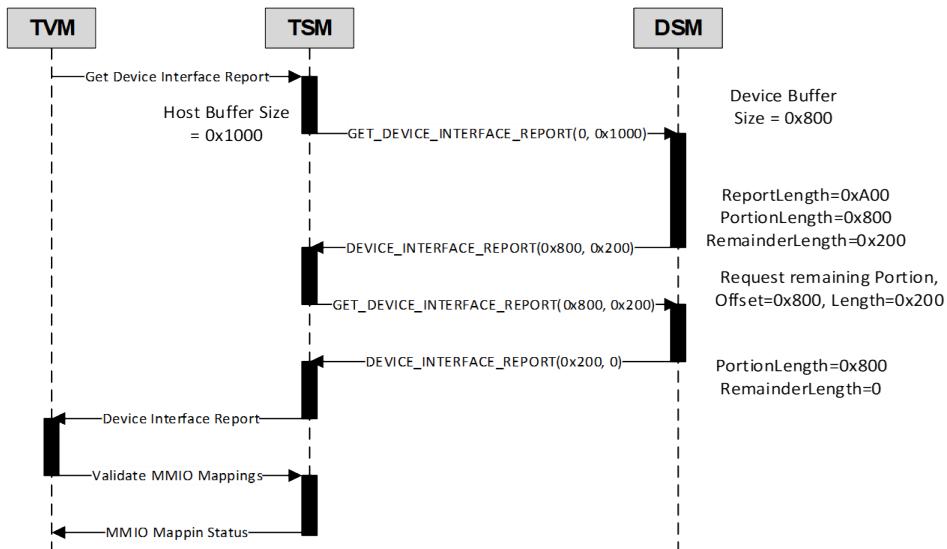


Figure 11-7 Example Flow Where DSM is Unable to Return Full Length Report

§ Table 11-17 defines the error codes and conditions:

Table 11-17 GET_DEVICE_INTERFACE_REPORT Error Response Codes

| Error Code | Description |
|-------------------------|----------------------------------|
| INVALID_REQUEST | OFFSET is invalid. |
| INVALID_INTERFACE_STATE | The TDI is not in CONFIG_LOCKED. |

11.3.12 GET_DEVICE_INTERFACE_STATE §

The GET_DEVICE_INTERFACE_STATE is used to request a DEVICE_INTERFACE_STATE from the device.

The device must fail the request if the following error is detected:

- Interface ID in the request is not hosted by the device.

11.3.13 DEVICE_INTERFACE_STATE §

Table 11-18 DEVICE_INTERFACE_STATE §

| Offset | Field | Size (Bytes) | Description |
|--|-----------|--------------|--|
| Payload (All fields in little endian format) | | | |
| 16 | TDI_STATE | 1 | TDI status 0 CONFIG_UNLOCKED 1 CONFIG_LOCKED 2 RUN 3 ERROR Others Reserved |

11.3.14 START_INTERFACE_REQUEST §

The START_INTERFACE_REQUEST carries the interface ID of the TDI. This request is used to transition the TDI to RUN, where is managed and operated by the TVM.

This request is expected to be generated by the TSM on request from the TVM.

The device must fail the request if any of the following errors are detected:

- If the interface ID in the request is not hosted by the device.
- START_INTERFACE_NONCE in the request is not valid i.e., does not match the nonce generated by the device in the LOCK_INTERFACE_RESPONSE.
- TDI is not in CONFIG_LOCKED.

If no errors are encountered, the device prepares to transition the referenced TDI to RUN. Moving the TDI to RUN may involve device side actions like enabling device side memory encryption, etc. The TDI must also invalidate the START_INTERFACE_NONCE before moving the TDI to RUN such that this nonce cannot be used again.

Table 11-19 START_INTERFACE_REQUEST §

| Offset | Field | Size (Bytes) | Description |
|--|-----------------------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | START_INTERFACE_NONCE | 32 | Device generated nonce for message (from LOCK_INTERFACE_RESPONSE) |

11.3.15 START_INTERFACE_RESPONSE §

§ Table 11-20 defines the error codes and conditions:

Table 11-20 START_INTERFACE_REQUEST Error Response Codes §

| Error Code | Description |
|---------------|---------------------------------|
| INVALID_NONCE | START_INTERFACE_NONCE mismatch. |

| Error Code | Description |
|-------------------------|----------------------------------|
| INVALID_INTERFACE_STATE | The TDI is not in CONFIG_LOCKED. |

11.3.16 STOP_INTERFACE_REQUEST §

The STOP_INTERFACE_REQUEST carries the interface ID of the TDI.

The device must fail the request the following error is detected:

- If the interface ID in the request is not hosted by the device.

In response to the STOP_INTERFACE_REQUEST the following actions must be performed:

- Abort all in-flight and accepted operations that are being performed by the TDI
- Wait for outstanding responses for the aborted operations
- All DMA read and write operations by the TDI are aborted or completed
- All interrupts from the TDI have been generated
- If function hosting the TDI is capable of Address Translation Service (ATS), all ATS requests by the TDI have completed or aborted. All translations cached in the device for ATS requests generated by this TDI have been invalidated.
- If function hosting the TDI is capable of Page Request Interface Service (PRI), no more page requests will be generated by the TDI. Additionally, either page responses have been received for all page requests generated by the TDI or the TDI will discard page responses for outstanding page requests.
- Scrub internal state of the device to remove secrets associated with the TDI such that those secrets will not be accessible.
- Reclaim and scrub private resources (e.g., memory encryption keys for device attached memories, etc.) assigned to the TDI.

The Device must generate the STOP_INTERFACE_RESPONSE once these actions are completed.

11.3.17 STOP_INTERFACE_RESPONSE §

No request-specific responses are defined

11.3.18 BIND_P2P_STREAM_REQUEST §

A TDI is permitted to support peer-to-peer transactions secured from end-to-end between two devices. Such devices must support configuring one or more selective IDE Stream(s) such that the selective IDE stream configuration registers provide the address and Requester ID ranges for the peer device. Such peer-to-peer IDE streams must be used by a device only if the device supports Address Translation Services and the capability is enabled for the device.

The BIND_P2P_STREAM_REQUEST binds such peer-to-peer stream IDs to the TDI. The device must fail the request if any of the following apply:

- Interface ID in the request is not hosted by the device
- TDI does not support binding peer-to-peer streams

Base 6.4 vs Base 6.3

- TDI is not in RUN
- $\Downarrow\text{Stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ specified does not have IDE keys programmed for all sub streams
- All IDE keys of the stream identified by the $\Downarrow\text{Stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ were not configured over the SPDM session on which the LOCK_INTERFACE_REQUEST was received
- Multiple IDE configuration registers have been programmed with the same $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$
- IDE configuration register for this stream is configured as the default stream
- Address and/or RID association registers of this streams IDE configuration registers overlap with other IDE configuration registers
- $\Updownarrow\text{If the XT Mode is supported, the XT Enable setting for the stream identified by the Stream ID is not matching the XT Enable configuration specified in the BIND_P2P_STREAM_REQUEST.}\Updownarrow$ ECN: Base 6.3 XT $\triangleleft\triangleright$

In response to the request, DSM carries out necessary actions on the device side to lock the IDE configurations for the specified $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ and enables mechanisms to track changes to the IDE configuration registers.

Through a device-specific mechanism the DSM ensures correctness of transaction ordering (i.e., if transactions were previously routed through the default $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ specified by the LOCK_INTERFACE_REQUEST then the TDI has implemented fences or other mechanism before it starts using the peer-to-peer $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ specified by this request).

Following processing the request, the device generates the BIND_P2P_STREAM_RESPONSE.

When a TDI generates a transaction, if the transaction's address or Requester ID as applicable matches an IDE configuration register and the $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ configured is one of the P2P streams bound to the TDI, then the device uses that P2P stream for the transaction. If the transaction does not match one of the P2P IDE streams, then the transaction uses the default stream identified by the $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ bound using the LOCK_INTERFACE_REQUEST.

All ATS requests and requests with addresses for which translations were not previously obtained from the Translation Agent in the Root Complex must use the default stream identified by the $\Downarrow\text{stream ID}\Downarrow \Updownarrow\text{Stream_ID}\Updownarrow$ bound at time of locking the TDI.

Table 11-21 BIND_P2P_STREAM_REQUEST §

| Offset | Field | Size (Bytes) | Description |
|---|--|------------------------------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | P2P_STREAM_ID | 1 | ID of the P2P stream to bind to this TDI. |
| $\Updownarrow\text{17}\Updownarrow$ ECN: Base 6.3 XT $\triangleleft\triangleright$ | $\Updownarrow\text{STREAM_CONFIG}\Updownarrow$ | $\Updownarrow\text{1}\Updownarrow$ | $\Updownarrow\text{Bit 0 XT_ENABLE_CONFIG}\Updownarrow$ $\Updownarrow\text{Bits 7:1 Reserved}\Updownarrow$ |

11.3.19 BIND_P2P_STREAM_RESPONSE §

§ Table 11-22 defines the error codes and conditions:

Table 11-22 BIND_P2P_STREAM_REQUEST Error Response Codes §

| Error Code | Description |
|-----------------|---|
| INVALID_REQUEST | TDI does not support binding P2P streams. |

| Error Code | Description |
|-------------------------|---|
| | <p>P2P_STREAM_ID is invalid</p> <p>Keys have not been configured for all sub-streams of the P2P stream</p> <p>Keys for the stream identified by P2P_STREAM_ID were not configured by the SPDM session on which the LOCK_INTERFACE_REQUEST was received</p> <p>IDE registers with configurations for the P2P_STREAM_ID is marked as the default stream.</p> <p>Multiple IDE registers are configured with the P2P_STREAM_ID</p> <p>The IDE registers configured for this P2P_STREAM_ID have overlaps with other valid IDE registers.</p> <p>If the XT Mode is supported, the XT Enable setting for the stream identified by the Stream ID is not matching the XT Enable configuration specified in the BIND_P2P_STREAM_REQUEST.^{↑↓}</p> |
| INVALID_INTERFACE_STATE | If the TDI is not in RUN. |

ECN: Base 6.3 XT[△]

11.3.20 UNBIND_P2P_STREAM_REQUEST §

The UNBIND_P2P_STREAM_REQUEST unbinds a previously bound peer-to-peer stream IDs from the TDI. The device must fail the request if any of the following apply:

- Interface ID in the request is not hosted by the device
- TDI does not support binding peer-to-peer streams
- TDI is not in RUN
- Stream ID^{↑↓} specified was not previously bound to this TDI

Following processing the request, the device generates the UNBIND_P2P_STREAM_RESPONSE.

An UNBIND_P2P_STREAM_RESPONSE implies that the device has successfully completed the following operations:

- All DMA read and write operations by the TDI using the specified P2P stream are aborted or completed
- Remove locking made active on the IDE configuration registers for this stream such that the IDE register may be reprogrammed without affecting the security of the TDI

If the device supports continuing the peer-to-peer operations following the unbind then through a device-specific mechanism the device ensures correctness of transaction ordering (i.e., if transactions were previously routed through this p2p stream then the TDI has implemented fences or other mechanism before it starts using the default Stream ID).^{↑↓}

Table 11-23 UNBIND_P2P_STREAM_REQUEST §

| Offset | Field | Size (Bytes) | Description |
|--|---------------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | P2P_STREAM_ID | 1 | ID of the P2P stream to unbind from this TDI. |

11.3.21 UNBIND_P2P_STREAM_RESPONSE §

§ Table 11-24 defines the error codes and conditions:

Table 11-24 UNBIND_P2P_STREAM_REQUEST Error Response Codes §

| Error Code | Description |
|-------------------------|--|
| INVALID_REQUEST | TDI does not support binding P2P streams. P2P_STREAM_ID is invalid P2P_STREAM_ID was not previously bound to this TDI. |
| INVALID_INTERFACE_STATE | If the TDI is not in RUN. |

11.3.22 SET_MMIO_ATTRIBUTE_REQUEST §

The SET_MMIO_ATTRIBUTE_REQUEST enables a TVM to update attributes of one or more MMIO ranges reported in the DEVICE_INTERFACE_REPORT. The MMIO ranges in a TDI that support updateable attributes are device specific.

The device must fail the request if any of the following apply:

- Interface ID in the request is not hosted by the device
- TDI does not support updateable MMIO attributes
- TDI does not support updateable MMIO attributes for the requested MMIO range
- TDI does not support the specified attribute for the requested MMIO range
- TDI does not support the value specified for the attribute
- TDI is not in RUN
- The MMIO range specified in the request is not associated with TDI

Responding with a failure is not fatal to the TDI and does not lead to a change in the TDI state.

Following processing the request, the device generates the SET_MMIO_ATTRIBUTE_RESPONSE.

IS_NON_TEE_MEM attribute may be updated to 1 to allow sharing the requested MMIO range with an entity not in the TVM trust boundary. Following the successful update of the attribute to ↑↓1,↓ ↑↑1:↑ ECN: Base 6.3 XT△↔

- ↑↑If the XT Mode is not supported or not enabled,↑ the specified MMIO range ↑↓may+↓ is permitted to↑ be accessed using ↑↓requests+ ↑↑a non-IDE Request or a Request+ with ↑↑the+ T bit ↑↓set to 0 or 1,↓ ↑↑Clear+ or ↑↑Set.↑ ECN: Base 6.3 XT△↔
- ↑↑If the XT Mode is enabled, the specified MMIO range is permitted to be accessed+ using a non-IDE Request.↓ ↑↑Request or a Request with the XT bit Set or Clear and the T bit Clear.↑ ECN: Base 6.3 XT△↔

While the processing of the request is outstanding, a device may continue to reject Requests ↑↓with the T bit Clear+ that access the MMIO range being ↑↓updated.↓ ↑↑updated:↑ ECN: Base 6.3 XT△↔

- ↑↑If the XT Mode is not supported or not enabled, Requests with the T bit Clear.↑ ECN: Base 6.3 XT△↔
- ↑↑If the XT Mode is enabled, Requests with the XT bit Set or Clear and the T bit Clear.↑ ECN: Base 6.3 XT△↔

IS_NON_TEE_MEM attribute may be updated to 0 to disallow sharing the MMIO range with an entity not in the TVM trust boundary. Following the successful update of the attribute to $\downarrow\downarrow 0, \uparrow\uparrow 0:\uparrow$ ECN: Base 6.3 XT $\triangleleft\triangleright$

- $\uparrow\uparrow$ If the XT Mode is not supported or not enabled, \uparrow the specified MMIO range $\uparrow\downarrow$ may \downarrow $\uparrow\uparrow$ must \uparrow only be accessed using Requests with $\uparrow\uparrow$ the \uparrow T bit $\downarrow\downarrow$ Set, and \downarrow $\uparrow\uparrow$ Set. \uparrow ECN: Base 6.3 XT $\triangleleft\triangleright$
- $\uparrow\uparrow$ If the XT Mode is enabled, the specified MMIO range must only be accessed using Requests with $\uparrow\uparrow$ the XT bit Set and the \uparrow T bit $\downarrow\downarrow$ Clear must be rejected. \downarrow $\uparrow\uparrow$ Set. \uparrow ECN: Base 6.3 XT $\triangleleft\triangleright$

While the processing of the request is outstanding, a device is permitted to continue to allow accesses $\uparrow\downarrow$ via \downarrow $\uparrow\uparrow$ via \uparrow ECN: Base 6.3 XT $\triangleleft\triangleright$

- $\uparrow\uparrow$ If the XT Mode is not supported or not enabled, \uparrow Requests with the T bit Clear. ECN: Base 6.3 XT $\triangleleft\triangleright$
- $\uparrow\uparrow$ If the XT Mode is enabled, Requests with the XT bit Set or Clear and the T bit Clear. \uparrow

A SET_MMIO_ATTRIBUTE_RESPONSE implies that the device has successfully completed updating the attributes for the specified MMIO range and the updated attributes are in affect for all subsequent accesses to this MMIO range.

Table 11-25 SET_MMIO_ATTRIBUTE_REQUEST §

| Offset | Field | Size (Bytes) | Description |
|--|------------|--------------|---|
| Payload (All fields in little endian format) | | | |
| 16 | MMIO_RANGE | 16 | <p>Base and size of the MMIO range to update attributes.</p> <ul style="list-style-type: none"> 8 bytes – First 4K page with offset added 4 bytes – Number of 4K pages in this range 4 bytes – Range Attributes <p>Bits 1:0 Reserved – must be zero.</p> <p>Bit 2 IS_NON_TEE_MEM – set to 1b if the range is non-TEE memory.</p> <p>Bits 15:3 Reserved</p> <p>Bits 31:6 Range ID – a device specific identifier for the specified range.</p> |

11.3.23 SET_MMIO_ATTRIBUTE_RESPONSE §

§ Table 11-26 defines the error codes and conditions:

Table 11-26 SET_MMIO_ATTRIBUTE_REQUEST Error Response Codes §

| Error Code | Description |
|-------------------------|---|
| INVALID_REQUEST | <p>TDI does not support updateable MMIO attributes</p> <p>TDI does not support updateable attributes for requested MMIO range</p> <p>TDI does not support specified attribute for requested MMIO range</p> <p>TDI does not support the value specified for the attribute</p> <p>The range specified in the request is not associated with TDI</p> |
| INVALID_INTERFACE_STATE | If the TDI is not in RUN. |

11.3.24 TDISP_ERROR §

The TDISP_ERROR is permitted to be used by the device to complete any of the requests issued to the device.

Table 11-27 TDISP_ERROR §

| Offset | Field | Size (Bytes) | Description |
|--|---------------------|--------------|----------------------|
| Payload (All fields in little endian format) | | | |
| 16 | ERROR_CODE | 4 | Error Code |
| 20 | ERROR_DATA | 4 | Error Data |
| 24 | EXTENDED_ERROR_DATA | Variable | Extended Error Data. |

Table 11-28 Error Code and Error Data §

| Error Code | Value (h) | Description | Error Data | Extended error data |
|------------------------------|-----------|---|-------------------------------|--|
| Reserved | 0000 | Reserved | Reserved | None |
| INVALID_REQUEST | 0001 | One or more request field is invalid. | Reserved | None |
| BUSY | 0003 | The Responder received the request message and the Responder decided to ignore the request message, but the Responder may be able to process the request message if the request message is sent again in the future. | Reserved | None |
| INVALID_INTERFACE_STATE | 0004 | The Responder received the request while in the wrong state, or received an unexpected request. For example, the GET_DEVICE_INTERFACE_REPORT before LOCK_INTERFACE_REQUEST, or any command between multiple GET_DEVICE_INTERFACE_REPORT | Reserved | None |
| UNSPECIFIED | 0005 | Unspecified error occurred. | Reserved | None |
| UNSUPPORTED_REQUEST | 0007 | Request code is unsupported | Request code | None |
| VERSION_MISMATCH | 0041 | The version is not supported | Reserved | None |
| VENDOR_SPECIFIC_ERROR | 00FF | Vendor defined | Length of extended error data | See required formatting of extended error data for vendor defined errors |
| INVALID_INTERFACE | 0101 | INTERFACE_ID does not exist. | Reserved | None |
| INVALID_NONCE | 0102 | The received nonce does not match the expected one. | Reserved | None |
| INSUFFICIENT_ENTROPY | 0103 | The Responder fails to generate nonce. | Reserved | None |
| INVALID_DEVICE_CONFIGURATION | 0104 | Invalid/Unsupported device configurations. | Reserved | None |

§ Table 11-29 defines the EXTENDED_ERROR_DATA format for vendor defined TDISP_ERROR response messages:

Table 11-29 EXTENDED_ERROR_DATA §

| Offset | Field | Size (Bytes) | Description |
|-------------------|-----------------|---------------|--|
| 0 | REGISTRY_ID | 1 | ID of the registry assigning the VENDOR_ID 00h – PCI-SIG assigned vendor ID 01h – CXL assigned vendor ID |
| 1 | VENDOR_ID_LEN | 1 | Length of VENDOR_ID field. |
| 2 | VENDOR_ID | VENDOR_ID_LEN | VENDOR_ID as assigned by the registry identified by REGISTRY_ID |
| 2 + VENDOR_ID_LEN | VENDOR_ERR_DATA | Variable | Vendor defined error data. |

11.3.25 VDM_REQUEST §

§ Table 11-30 defines the VDM_REQUEST format:

Table 11-30 VDM_REQUEST §

| Offset | Field | Size (Bytes) | Description |
|-------------------|---------------|---------------|--|
| 0 | REGISTRY_ID | 1 | ID of the registry assigning the VENDOR_ID 00h – PCI-SIG assigned vendor ID 01h – CXL assigned vendor ID |
| 1 | VENDOR_ID_LEN | 1 | Length of VENDOR_ID field. |
| 2 | VENDOR_ID | VENDOR_ID_LEN | VENDOR_ID as assigned by the registry identified by REGISTRY_ID |
| 2 + VENDOR_ID_LEN | VENDOR_DATA | Variable | Vendor defined data. |

11.3.26 VDM_RESPONSE §

§ Table 11-31 defines the VDM_RESPONSE format:

Table 11-31 VDM_RESPONSE §

| Offset | Field | Size (Bytes) | Description |
|-------------------|---------------|---------------|--|
| 0 | REGISTRY_ID | 1 | ID of the registry assigning the VENDOR_ID 00h – PCI-SIG assigned vendor ID 01h – CXL assigned vendor ID |
| 1 | VENDOR_ID_LEN | 1 | Length of VENDOR_ID field. |
| 2 | VENDOR_ID | VENDOR_ID_LEN | VENDOR_ID as assigned by the registry identified by REGISTRY_ID |
| 2 + VENDOR_ID_LEN | VENDOR_DATA | Variable | Vendor defined data. |

11.3.27 $\uparrow\downarrow$ SET_TDISP_CONFIG_REQUEST \uparrow §

ECN: Base 6.3
XT \triangle \triangleleft \triangleright

$\uparrow\downarrow$ The SET_TDISP_CONFIG_REQUEST is used by the TSM to configure TDISP settings on the device. All TDIs must be in the CONFIG_UNLOCKED state for the device to process this request. \uparrow

- $\uparrow\downarrow$ XT_MODE_ENABLE – When Set, enables the TDIs on the device to operate in the XT Mode. \uparrow
- $\uparrow\downarrow$ XT_BIT_FOR_LOCKED_MSIX – When the XT Mode is enabled, indicates the value of the XT bit to be used when generating Untranslated Memory Requests (or ATS Translation Requests) associated with the locked MSI-X table. \uparrow
 - $\uparrow\downarrow$ This bit must be Clear if the XT_MODE_ENABLE is Clear. \uparrow

Table 11-32 $\uparrow\downarrow$ SET_TDISP_CONFIG_REQUEST \uparrow §

| $\uparrow\downarrow$ Offset \uparrow | $\uparrow\downarrow$ Field \uparrow | $\uparrow\downarrow$ Size (Bytes) \uparrow | $\uparrow\downarrow$ Description \uparrow |
|--|--|--|---|
| $\uparrow\downarrow$ Payload (All fields in little endian format) \uparrow | | | |
| $\uparrow\downarrow$ 16 \uparrow | $\uparrow\downarrow$ DSM_CONFIG \uparrow | $\uparrow\downarrow$ 4 \uparrow | <p>$\uparrow\downarrow$Bit 0 XT_MODE_ENABLE\uparrow</p> <p>$\uparrow\downarrow$Bit 1 XT_BIT_FOR_LOCKED_MSIX\uparrow</p> <p>$\uparrow\downarrow$Bits 31:2 Reserved\uparrow</p> |

11.3.28 $\uparrow\downarrow$ SET_TDISP_CONFIG_RESPONSE \uparrow §

ECN: Base 6.3
XT \triangle \triangleleft \triangleright

$\uparrow\downarrow$ \$ Table 11-33 defines the error codes and conditions: \uparrow

Table 11-33 $\uparrow\downarrow$ SET_TDISP_CONFIG_RESPONSE \uparrow §

| $\uparrow\downarrow$ Error Code \uparrow | $\uparrow\downarrow$ Description \uparrow |
|---|--|
| $\uparrow\downarrow$ INVALID_REQUEST \uparrow | <p>$\uparrow\downarrow$XT_MODE_ENABLE is Set, but the XT Mode is not supported by the device.\uparrow</p> <p>$\uparrow\downarrow$XT_MODE_ENABLE is Clear, but XT_BIT_FOR_LOCKED_MSIX is Set.\uparrow</p> |
| $\uparrow\downarrow$ INVALID_INTERFACE_STATE \uparrow | $\uparrow\downarrow$ One or more TDIs is not in CONFIG_UNLOCKED \uparrow |

11.4 Device Security Requirements §

11.4.1 Device Identity and Authentication §

A TEE-I/O capable device must implement the [SPDM] as the device secure communication protocol with the host. The device must use SPDM protocol to report the device identity and support the authentication. The security property defined in SPDM specification must be satisfied.

The device is recommended to implement the Device Identifier Composition Engine (DICE) architecture specified by the Trusted Computing Group (TCG). In this case, a DICE certificate must be returned in SPDM protocol and used to provide device identity and support authentication.

11.4.2 Firmware and Configuration Measurements §

A TEE-I/O capable device must implement [SPDM] to return device measurements to the TSM. The TEE-I/O device may report hash-based measurement(s), and/or secure version number (SVN) to the host.

The device is recommended to implement TCG DICE. In this case, the device hash-based measurement and/or SVN must also be included in the DICE TCB info structure. The information reported in SPDM Measurement response and DICE TCB info must be consistent.

The device is permitted to support mutable firmware update. In this case, the device is recommended to follow established industry firmware resilience guidelines, such as NIST SP 800-193, to ensure the new firmware provides equal or better security.

The device is permitted to support runtime update without reset. Such a capability must be reported via INTERFACE_INFO, and can be blocked via NO_FW_UPDATE. A runtime update image must be of an equal or higher SVN if active TDIs are to be maintained. The device must report SPDM MEAS_FRESH_CAP to indicate if the device has capability to report fresh measurements or old measurements computed during last device reset. If a DICE device supports runtime update, the security property defined in DICE specification must still be satisfied, such as DICE certificate creation. An attempt to lower the SVN must be rejected by the device if there are active TDIs in CONFIG_LOCKED or RUN. Alternately, if the device has the capability to secure and clean all TVM data in a trusted manner before such a downgrade, the device is permitted to transition the interfaces in CONFIG_LOCKED or RUN to ERROR, terminate IDE streams used for TEE-I/O, and terminate the SPDM session with TSM before a firmware downgrade.

Devices that do not support runtime update when there are TDIs in CONFIG_LOCKED or RUN must handle a forced update by terminating the IDE streams used for TEE-I/O, and the SPDM session between the TSM and DSM, and transitioning associated TDI(s) to ERROR.

11.4.3 Securing Interconnects §

TEE-I/O capable devices must support Integrity and Data Encryption (IDE) to protect transactions on the interconnect between the device and the Root Complex. Devices must support selective IDE streams between the Root Complex and the device. Use of IDE to protect transactions may not be required for RCiEP.

Peer-to-peer links, between the devices must be protected to avoid loss of confidentiality and integrity. Peer-to-peer must use IDE to secure communications. Other kinds of interconnects must be protected using interconnect specific extensions such that they provide equivalent security to address the threat model outlined in IDE.

The symmetric stream encryption keys and IV of each IDE Sub-Stream are secret and compromising these breaks the security of the solution. The device must implement adequate security measures to prevent leakage of the encryption key at rest and in use. The stream encryption keys must not be revealed in plaintext form outside the device. The device must not allow modifications to the stream encryption keys or the IV through untrusted mechanisms.

TEE-I/O capable devices must transition interfaces CONFIG_LOCKED, RUN to ERROR when the IDE stream(s) bound to those interface transition to Insecure state. The device must implement suitable mechanisms to contain propagation of data past the transition to Insecure state.

Receipt of a Completion with UR/CA or Completion timeout (following recovery retries) for requests initiated by a device on an interface in CONFIG_LOCKED, RUN indicates occurrence of an uncorrectable error (e.g., IOMMU translation tables corrupted, etc.) in handling the ~~↑↓non-posted request.~~ ↑↓Non-Posted Request or UIO Request. The device must transition the interface for which UR/CA was received to ERROR to stop consumption and propagation of errors if the error is not recoverable.

 Errata: Base 6.3
B834△◀

11.4.4 Device Attached Memory §

Certain devices implement device attached memory where such memory is used by logic in the device to host the TVM data. The device must ensure the confidentiality of the TVM data stored in such memory devices such that the TVM data is not revealed as plaintext outside the device or to entities not in the TVM TCB. To the maximum extent possible the ciphertext associated with the TVM data must not be exposed outside the device. The device may additionally provide integrity properties on the TVM data.

IMPLEMENTATION NOTE: SECURITY OF DEVICE ATTACHED MEMORY §

Certain devices may implement memory encryption as a mechanism to provide confidentiality of TVM data stored into those memory devices. Such devices may additionally provide integrity properties on the memory content. The configurations of memory encryption as well as other configurations related to the device attached memory in such devices is managed by the DSM. The TSM relies on the DSM to secure such configurations. Securing of memory configurations, establishing memory encryption, and all related security checks must be performed no later than in response to the first LOCK_INTERFACE_REQUEST received by the DSM for a device interface hosted by the device. If the memory configurations are not acceptable to meet TVM security requirements, then device interfaces must not be transitioned to CONFIG_LOCKED.

11.4.5 TDI Security §

TEE-I/O capable devices must support [Secure SPDM] to establish a secure communication session between the TSM and DSM. The devices must support the TDI state and the device interface management protocol in TDISP for managing the device security states as they are assigned to TVMs and detached from TVMs.

All sub-streams of the IDE stream bound to the TDI must be programmed over the same SPDM session used to lock the interface. Attempting to configure IDE keys into a sub-stream using different SPDM sessions is an error and must be rejected. IDE key refresh must be accepted only on the SPDM session that was used to establish the initial IDE keys.

TDIs in RUN state may transmit/receive transactions with peer TDIs over peer-to-peer selective IDE streams if the device model supports such communication. The peer-to-peer selective IDE streams may be used for communication if the IDE keys for such streams were also configured with the same SPDM session as that was used to transition the TDI to CONFIG_LOCKED.

When the SPDM session used to program an IDE stream key enters session termination phase, all IDE streams configured with keys over that SPDM session must transition to an Insecure state and all TDI that were transitioned to CONFIG_LOCKED over that SPDM session must transition to ERROR.

When an IDE stream transitions to Insecure state, all TDIs in CONFIG_LOCKED/RUN bound to that IDE stream must transition to ERROR.

TEE-I/O capable devices must ensure that confidentiality and integrity of configurations and data associated with a TVM assigned TDI. Devices must implement suitable mechanisms to prevent leakage and tamper of such configuration and data from other TDIs including from the physical function of the device.

Devices may consider the administration functions provided through the physical function or the administrative queue of the device as untrusted. Typically, such administrative interfaces managed by the VMM are trusted by the TDIs (such as virtual function or TDIs) by virtual machines that include the VMM in the trust boundary. As the TVM may not include the VMM in the trust boundary, the administrative actions performed by the VMM may need to be mediated by the DSM

to ensure that the administrative actions do not compromise the confidentiality or integrity of the TVM data, link encryption keys, SPDM session keys, and other secrets that are essential to the security of the solution. Administrative functions (e.g., QOS configurations, etc.) that are benign and could only lead to denial of service if misused may be allowed when the TDI is in CONFIG_LOCKED, RUN state.

TEE-I/O capable device should be designed to avoid or minimize the need for host driver intervention for TDI specific configuration or control operations where such operations if carried out maliciously may lead to confidentiality or integrity of the TVM data being compromised. Certain operations that only affect the quality of service or performance characteristics of the TDI but otherwise are benign to the functional correctness and security of the interface may be carried out by host driver on behalf of the TVM. The host driver may not be in the trust boundary of the TVM assigned TDIs and may not be trusted to provide software emulation of TVM operations.

Changing the device configurations such as reprogramming the physical function or virtual function BAR, changing configurations of the TDIs, etc. may be used to temporarily drop a transaction originated by the TVM or lead to exposure of TVM confidential data (e.g., redirecting it to registers of a different virtual function than the one to which it is bound, etc.). Such configurations elements may be in the configuration space of the physical function, configuration space of the virtual functions, MMIO registers mapped by the physical function or the virtual function BARs, etc. Such configurations changes should be considered hostile actions and the TDI must be transition from CONFIG_LOCKED/RUN to ERROR.

Errors or failures encountered in the DSM or in other parts of logic in the device where the failures are not recoverable, or lead to the DSM losing state of the TDI, must lead to the device transitioning to a secure failed state where the TDIs in CONFIG_LOCKED/RUN transition to ERROR.

11.4.6 Data Integrity Errors §

Receipt of a poisoned TLP on an interface in RUN indicates occurrence of uncorrectable data integrity errors. Except when the TDI implements mechanisms to recover from and/or suitably handle, receipt of such a TLP must transition the interface from RUN to ERROR to prevent bad data consumption and propagation. The device should implement suitable protection schemes such as parity or ECC on its internal data buffers and caches to detect data integrity errors. If uncorrectable data integrity errors were detected, then the affected interfaces must transition from CONFIG_LOCKED/RUN to ERROR and poison signaled to the requester as appropriate. The device may provide a mechanism to report and log the occurrence of such errors. Devices must scrub and clear information in such logs and reporting registers (e.g., syndrome) that may reveal confidential data.

11.4.7 Debug Modes §

Devices may support multiple debug modes or debug capabilities. Some of the debug capabilities may allow a software debugger to collect statistics, error logs, etc. Such debug capabilities must not affect the security of the device, and must not lead to a compromise of the confidentiality or integrity of the TVM data provided to the device.

Debug configuration may, for example, be reported in SPDM measurements, such as non-invasive debug mode or invasive debug mode. A DICE device may report operation flags in DICE TCB info, such as debug mode.

Other debug modes may allow the debugger to affect the security of the device by providing mechanisms, for example, to bypass signature verification, trace data flowing through the device buses, affect the measurement process itself, etc. The device may support and use debug-mode identity certificates to identify such debug modes being active. This enables the TVM (and remote verifiers) to determine if the TVM may provide secrets to the device. TEE-I/O capable devices may restrict authorization of such debug modes to an early window following cold reset (i.e., the debug authorization occurs before secrets have been provided to the device or secrets in the device itself have been unlocked). When the debug authorization window is active, the device must not participate in SPDM session setup. Alternately, a TEE-I/O capable device may allow debug authorization to always occur but in response to the debug authorization request transition the IDE stream states to Insecure, terminate the SPDM session with the TSM, and transition all TDIs in

CONFIG_LOCKED, RUN state to ERROR state prior to enabling the debug interface. Such devices provide the assurance that secrets already provided to the device by the TVM are not accessible to the debugger.

11.4.8 Conventional Reset §

A conventional reset (cold, warm, or hot) leads to the device changing all its Port registers and state machines to their initialization values, and the TDISP state of all TDIs transitions to CONFIG_UNLOCKED.

Device reset architecture must ensure that all TVM data, IDE keys, other encryption keys (e.g., P2P links, intra-device interconnects, etc.) and SPDM session keys are cleared such that they are not exposed in plaintext through any mechanism following exit from the reset.

Devices may authorize debug following a reset and if the stream encryption keys are not scrubbed and cleared, they may become accessible to the debugger using debug tools. In addition to the stream encryption keys, TVM data held in clear text in the device (e.g., in the device's coherent caches, register files, SRAMs, etc.) may become accessible to debuggers using debug tools. The TEE-I/O capable device must implement suitable mechanisms to scrub residual secrets from device internal structures before authorizing debug access following a reset.

Devices may lose power as part of the conventional reset and may not have state coming out of the reset to determine whether the device was in use by a TVM or not. A TEE-I/O capable device should assume that prior to a conventional reset there may have been a TDI associated with a TVM and thus implement suitable mechanisms to ensure that residual data and secrets are not leaked in plaintext following such a reset.

The device measurement registers must be reset to their default values as part of a reset that requires firmware reload. Some devices may reload firmware on a warm reset, whereas others may require a cold reset or a D3 state transition.

11.4.9 Function Level Reset §

A function level reset of a VF or non-IOV function must affect the TDI hosted by that function. A function level reset of the PF must affect all subordinate VF TDIs. A function level reset must transition all affected TDIs from CONFIG_LOCKED, RUN state to ERROR state such that a STOP_INTERFACE_REQUEST request is required to clean up the TDI state and scrub TVM data/secrets prior to the transition of the affected TDIs to CONFIG_UNLOCKED state.

A functional level reset to Functions other than Function 0 does not affect active SPDM sessions or IDE streams.

11.4.10 Address Translation Services (ATS) and Access Control Services (ACS) §

TEE-I/O capable devices ↑↑must enforce integrity of the Address Translation Cache (ATC) such that the translations provided by the Root Complex cannot be modified through untrusted accesses. TDIs↑ that support ↑↑both↑ ATS and ↑↓have ATS enabled↓ ↑↑XT Mode↑ must ↑↓generate translation requests and page requests for a TDI in RUN state with T↑ ↑↑have the TEE Exclusive Memory Attribute Supported↑ bit ↑↓Set.↓ ↑↑Set in the ATS Capability Register.↑ ECN: Base 6.3 XT△↔
↑↓If device supports IDE, then these requests↓

↑↑Translation Requests↑ must be generated over the IDE stream bound to the TDI by the LOCK_INTERFACE_REQUEST. ↑↓ATS requests↓ ↑↑Translation Requests↑ not sent using ↑↓the↓ ↑↑this↑ default IDE stream, must not be assumed to accurately reflect the permissions of that TDI, and as such if the device is caching host-memory, must not allow sharing between TEE TDIs and non-TEE TDIs of that cached value. ECN: Base 6.3 XT△↔
↑↑When the XT Mode is enabled, it is allowed for the device to share that cached value between TEE TDIs and non-TEE TDIs if the cached value corresponds to the non-TEE memory (TEE Exclusive Memory Attribute (TE bit) is Clear).↑
↑↓ATS↓

Base 6.4 vs Base 6.3

If the XT Mode is not supported or not enabled, the following rules apply to Translation Requests issued by the TDI while and the corresponding Translation Completions:

ECN: Base 6.3 XT△↔

- Translation Requests (other than those using the address associated with MSI/MSI-X capability) issued by the TDI in RUN must have Set the T bit.**
- For the Translation Requests issued by the TDI in RUN, the corresponding Translation Completion(s) must be handled normally if and only if the TDI is still in RUN, and must otherwise be rejected.**
 - It is permitted for the TDI to ignore the value of the T bit Set, in Translation Completion(s), or to transition to ERROR if the value of the T bit in Translation Completion(s) is not matching the Translation Request.**

If the XT Mode is enabled, the following rules apply to Translation Requests issued by the TDI and the corresponding Translation Completions:

ECN: Base 6.3 XT△↔

- Translation Requests (other than those using the address associated with MSI/MSI-X capability) issued by the TDI in RUN:**
 - If the access must be restricted to TEE memory, TDI must Set the XT bit and Set the T bit.**
 - If the access must be restricted to non-TEE memory, TDI must Set the XT bit and Clear the T bit.**
 - If there are no specific restrictions determined by the TDI for the access, TDI must Clear the XT bit and Set the T bit.**
- For the Translation Requests issued by the TDI in RUN, the corresponding Translation Completion(s) must be received handled normally if and only if the TDI is still in RUN, and must otherwise be rejected.**
 - It is permitted for the TDI to ignore the value of the XT bit and the T bit in Translation Completion(s), or to transition to ERROR if the values of the XT bit and the T bit in Translation Completion(s) are not matching the Translation Request.**

Translation Requests issued by the TDI not in RUN must Clear the XT bit and Clear the T bit.

ECN: Base 6.3 XT△↔

If the XT Mode is not supported or not enabled, TA must return Translation Completion(s) with the matching T bit Set, value as the Translation Request. If the XT Mode is enabled, TA must return Translation Completion(s) received with the matching XT bit and T bit values as the Translation Request.

ECN: Base 6.3 XT△↔

If the TEE Exclusive Memory Attribute Supported bit is Clear in the ATS Capability Register, TDI must transition ignore the TEE Exclusive Memory Attribute (TE bit) received in the Translation Completions.

ECN: Base 6.3 XT△↔

Translated Requests must be generated over the IDE stream(s) bound to the TDI. When there are no P2P streams bound to the TDI, all Translated Requests must be directed upstream towards the Root Complex over the IDE stream bound to the TDI by the LOCK_INTERFACE_REQUEST. When there are P2P streams bound to the TDI, the TDI Translated Read Requests are permitted to be sent over a P2P stream bound to the TDI using the BIND_P2P_STREAM_REQUEST.

ECN: Base 6.3 XT△↔

If the XT Mode is not supported or Write not enabled, the following rules apply to Translated Requests issued by the TDI and the corresponding Completions:

ECN: Base 6.3 XT△↔

- Translated Requests issued by the TDI in RUN must Set the T bit.**
- For Translated Requests issued by the TDI while in RUN, the corresponding Completion(s) must be handled normally if and only if the TDI is still in RUN, and must otherwise be rejected.**
 - TDI must ignore the value of the T bit in Received Completions.**

If the XT Mode is enabled, the following rules apply to Translated Requests issued by the TDI and the corresponding Completions:

ECN: Base 6.3 XT△↔

- For Translated Requests issued by the TDI while in RUN, the TDI must configure the XT bit and the T bit values using the TEE Exclusive Memory Attribute (TE bit) provided by the Translation Agent (TA) for the memory location being accessed:
 - If the TE bit is Set, TDI must Set the XT bit and Set the T bit. Completions for ATS.
 - If the TE bit is Clear, TDI must Set the XT bit and Clear the T bit.
- Translated Read Requests issued by the TDI while in RUN are permitted to RUN, the corresponding Completion(s) must be handled normally if and only if the TDI is still in RUN, and must otherwise be received with rejected.
 - TDI must ignore the value of the XT bit and the T bit Set or Clear in Received Completions.

Translated Requests issued by the TDI not in RUN must Clear the XT bit and Clear the T bit.

ECN: Base 6.3 XT△↔

ATS Invalidate

Page Request and Invalidate Completion Messages are permitted to be generated over the IDE stream bound to use or not use IDE. If the TDI by the LOCK_INTERFACE_REQUEST, and the PRG Response Message(s) must be received on the same IDE stream.

ECN: Base 6.3 XT△↔

If the XT Mode is used, Invalidate Requests applying to translations for a Page Requests issued by the TDI and the corresponding PRG Responses:

ECN: Base 6.3 XT△↔

- Page Requests issued by the TDI in RUN must Set the XT bit. Page Requests from multiple interfaces in RUN state are permitted to have be grouped into a Page Request Group.
- For the T bit Set or Clear. If Page Requests issued by the Invalidate Request uses IDE, then TDI while in RUN, the Invalidate Completion corresponding PRG Response(s) must use be handled normally if and only if all of the same IDE Stream as following conditions are satisfied:
 - the Invalidate Request, and must match TDI is in RUN
 - the T bit in the PRG Response is matching the T bit value from of the Invalidate Request. Page Request Messages from Request; otherwise, the PRG Response must be rejected.

If the XT Mode is enabled, the following rules apply to Page Requests issued while in RUN, by the TDI and the corresponding PRG Responses:

ECN: Base 6.3 XT△↔

- Page Requests issued by the TDI in RUN must Clear the XT bit and Set the T bit. Page requests Requests from multiple interfaces in RUN are permitted to be grouped into a Page Request Group.
- For the Page Requests issued by the TDI while in RUN, the corresponding PRG Response(s) must use be handled normally if and only if all of the same IDE Stream as following conditions are satisfied:
 - the corresponding Page Request, TDI is in RUN
 - the XT bit and must have the T bit Set. A violation of this rule must result in the TDI transitioning to ERROR. Untranslated P2P requests generated PRG Response are matching the XT bit and the T bit values of the Page Request; otherwise, the PRG Response must be rejected.

↑↑↑Page Requests issued by ↑↓a↑ ↑↑the↑ TDI ↑↑not↑ in RUN ↑↓state↑ must ↑↓be redirected upstream towards↑ ↑↑Clear↑ the ↑↓Root Complex over↑ ↑↑XT bit and Clear↑ the ↑↓default↑ ↑↑T bit.↑

ECN: Base 6.3 XT△↔

↑↑↑Invalidate Request Messages and Invalidate Completion Messages are permitted to use or not use IDE. If the Invalidate Request Message uses IDE, then the Invalidate Completion Message must use the same↑ IDE stream ↑↓bound to↓ ↑as↑ the ↑↓TDI when↓ ↑↑Invalidate Request Message.↑

ECN: Base 6.3 XT△↔

↑↑↑If↑ the ↑↓interface was transitioned↓ ↑↑XT Mode is not supported or not enabled, the following rules apply↑ to ↑↓CONFIG_LOCKED state.↑ ↑↑Invalidation Requests and corresponding Invalidations Completions:↑

ECN: Base 6.3 XT△↔

↑↓When there↓

- ↑↑↑Invalidation Requests applying to translations for a TDI in the RUN state↑ are ↑↓no P2P streams bound↓ ↑↑permitted↑ to ↑↑have↑ the ↑↓interface, all translated P2P requests↓ ↑↑T bit Set or Clear.↑**
- ↑↑↑Invalidation Completion↑ generated by ↑↓a↑ TDI must ↑↓be directed upstream towards↑ the ↑↓Root Complex over↓ ↑↑same T bit value as↑ the ↑↓default IDE stream bound to↓ ↑↑Invalidation Request.↑**

ECN: Base 6.3 XT△↔

↑↑↑If↑ the ↑↓interface when↓ ↑↑XT Mode is enabled,↑ the ↑↓TDI was transitioned↓ ↑↑following rules apply↑ to ↑↓CONFIG_LOCKED state. When there are P2P streams bound↓ ↑↑Invalidation Requests and corresponding Invalidations Completions:↑

ECN: Base 6.3 XT△↔

- ↑↑↑Invalidation Requests applying↑ to ↑↓the TDI, translated P2P requests generated by↓ ↑↑translations for↑ a TDI ↑↓may be sent over a P2P stream bound↓ ↑↑in the RUN state are permitted↑ to ↑↑have↑ the ↑↓TDI using↓ ↑↑XT bit Set or Clear and↑ the ↑↓BIND_P2P_STREAM_REQUEST.TEE I/O capable devices↓ ↑↑T bit Set or Clear.↑**
- ↑↑↑Invalidation Completion generated by TDI↑ must ↑↓enforce integrity of the Address Translation Cache (ATC) such that↓ ↑↑use↑ the ↑↓translations provided by↓ ↑↑same XT bit and T bit values as↑ the ↑↓Root Complex cannot be modified through untrusted accesses.↓ ↑↑Invalidation Request.↑**

ECN: Base 6.3 XT△↔

Requirements on the device's PASID/ATS capabilities:

- Execute Permission Supported** must be Clear
- Global Invalidate Supported** must be Clear

↑↓Devices must, in all Translation Completions, treat the G bit as zero.↑ ↑↑↑

ECN: Base 6.3 XT△↔

The use of Access Control Services (ACS) mechanisms for redirection must be coordinated with the device configuration to ensure that the correct Selective IDE Stream will be used. Specifically:

- ACS Translation Blocking
- ACS P2P Request Redirect
- ACS P2P Completion Redirect
- ACS Upstream Forwarding
- ACS P2P Egress Control
- ACS Direct Translated P2P

11.5 Requirements Placed on Host Security due to TDI Requirements §

11.5.1 Address Translation §

The property of memory being either TEE memory or non-TEE memory, must, as observed by a TVM executing on the host, match the view of memory as observed by a TDI assigned to that TVM. The translation agent (TA) is permitted to use the T bit being 1 ↑↑or, when the XT Mode is enabled, the T bit and/or the XT bit being 1,↑ to identify requests originated by a TDI in RUN state. The TVM relies on the TSM and TA to translate requests from TVM assigned TDIs such that:

ECN: Base 6.3 XT△↔

1. An untrusted VMM cannot compromise the integrity of translation of Guest Physical Address or Guest Virtual Address to a physical address.
2. The Translation Agent (TA) ensures that Untranslated Requests transmitted by a TDI are checked against the access permissions for that TDI.
3. An Address Translation Cache (ATC) in the TDI, if enabled, is consistent with the Address Translation and Protection Tables (ATPT) associated with the TVM and changes to the ATPT are observed by both the CPU TLB, IOMMU TLB, as well as the ATC.
4. Identifiers such as Requester ID (and PASID) used for ATPT lookup are verified for integrity such that attempts to forge these identifiers are prevented.
5. Identifiers such as Requester ID and ITAGs used to process Invalidate Completions are verified for integrity such that attempts to forge these identifiers are prevented.
6. ↑↑When the XT Mode is enabled, TA performs the access checks based on both the T bit and the XT bit received on the Request.↑ ECN: Base 6.3 XT△↔
7. ↑↑When the XT Mode is enabled, TA must return the TEE Exclusive Memory Attribute (TE bit) associated with the Translated Addresses upon a successful translation. TA is permitted to return this TEE Exclusive Memory Attribute (TE bit) to any TEE-IO capable device, including when the XT Mode is not supported or not enabled.↑ ECN: Base 6.3 XT△↔

11.5.2 MMIO Access Control §

The TSM must provide the following access controls on TVM assigned MMIO resources:

1. Verify that all MMIO resources reported by the device through the DEVICE_INTERFACE_REPORT have been made accessible to the TVM and the order in which these resources are mapped into the TVM address space match the expected mapping order.
2. Restrict a TVM and TVM assigned TDIs to only access MMIO resources that have been assigned to that TVM.
3. ↑↑If the XT Mode is not supported or not enabled,↑ TLPs with the T bit Set may be generated to MMIO resources of a TVM only by accesses originated by that TVM or by components in the TCB of that TVM such as other TDIs assigned to that TVM that are in RUN state.
 - ↑↑Memory Requests targeting TEE memory must have the T bit Set.↑
 - ↑↑Memory Requests targeting non-TEE memory may have the T bit Clear or Set.↑
4. ↑↑If the XT Mode is enabled, TLPs with the XT bit and/or the T bit Set may be generated to MMIO resources of a TVM only by accesses originated by that TVM or by components in the TCB of that TVM such as other TDIs assigned to that TVM that are in RUN state.↑ ECN: Base 6.3 XT△↔

- ~~↑↑Memory Requests targeting TEE memory must have the XT bit Set and the T bit Set.~~
- ~~↑↑Memory Requests targeting non-TEE memory may have the XT bit Set or Clear and must have the T bit Clear.~~
- ~~↑↑Memory Requests must not be generated with the XT bit Clear and the T bit Set.~~

11.5.3 DMA Access Control §

The TSM must provide the following access controls to TVM assigned memory and MMIO resources:

1. DMA through untranslated or already-translated requests to TVM memory must be allowed only from TDIs accepted by a TVM in its TCB. Such DMA may be to the confidential memory of the TVM or non-confidential memory accessible to the TVM. When a TVM accepts a TDI in its TCB, it accepts the entirety of the device in its TCB. A device is in the TCB of all TVMs that accept TDIs of that device in their TCBs. The TVM that accepts a device into its TCB trusts the device to not spoof identifiers used for DMA access control such as the source Requester ID (and the PASID). A TVM that does not accept a TDI of a device must not have the device in its TCB and such devices must not have access to TVM memory or MMIO resources using either untranslated or already-translated requests.
2. ~~↑↑If the XT Mode is not supported or not enabled:~~
 - ~~↑↑Untranslated Memory Requests, ATS Translation Requests, and ATS Translated Requests with the T bit Clear must be restricted to accessing non-TEE memory.~~
 - ~~↑↑Untranslated Memory Requests and ATS Translation Requests with the T bit Set must be restricted to accessing TEE memory or non-TEE memory depending upon the address translation performed by the TA.~~
 - ~~↑↑ATS Translated Requests with the T bit Set must be restricted to accessing TEE memory or non-TEE memory depending upon the TEE access checks performed by the TA.~~ECN: Base 6.3 XT
3. ~~↑↑If the XT Mode is enabled:~~
 - ~~↑↑Untranslated Memory Requests, ATS Translation Requests, and ATS Translated Requests with the XT bit Clear and the T bit Clear must be restricted to accessing non-TEE memory.~~
 - ~~↑↑Untranslated Memory Requests and ATS Translation Requests with the XT bit Clear and the T bit Set must be restricted to accessing TEE memory or non-TEE memory depending upon the address translation performed by the TA.~~
 - ~~↑↑ATS Translated Requests with the XT bit Clear and the T bit Set must be rejected. Untranslated Memory Requests, ATS Translation Requests, and ATS Translated Requests with the XT bit Set and the T bit Set must be restricted to accessing TEE memory.~~
 - ~~↑↑Untranslated Memory Requests, ATS Translation Requests, and ATS Translated Requests with the XT bit Set and the T bit Clear must be restricted to accessing non-TEE memory.~~ECN: Base 6.3 XT

11.5.4 Device Binding §

The TVM uses the authentication and measurement reporting protocol specified by [SPDM] to determine if the identity and measurements reported by the device hosting the TDI are acceptable prior to admitting the device into its TCB. The TVM needs to further determine if the authenticated device is presently bound to the host using an IDE stream (if applicable) and ~~↑↓A↓ ↑↑An↑~~ ~~↑↑An~~ SPDM session established by the TSM.

The TSM must provide a trusted mechanism to determine if:

1. ~~↑↓A↓ ↑↑An↑~~ ~~↑↑An~~ SPDM session is active between the TSM and the DSM in the device authenticated by the TVM.

2. IDE keys for the IDE stream used by that TDI have been established by the TSM .

11.5.5 Securing Interconnects §

The symmetric stream encryption keys and IV of each IDE Sub-Stream are secret and compromising these breaks the security of the solution. The host must implement adequate security measures to prevent leakage of the encryption key at rest and in use. The stream encryption keys and IV must not be revealed outside the TSM and the TSM TCB. The host must not allow modifications to the stream encryption keys or the IV through untrusted mechanisms. These protections apply to all host-specific mechanisms that contribute to maintaining confidentiality and integrity of IDE streams, for example Key Set change mechanisms or key refresh timers.

The host must program a unique encryption key for each IDE sub-stream and must not ~~↑live-use↓~~ ↑reuse↑ the encryption keys when the IDE sub-stream keys are refreshed.

11.5.6 Data Integrity Errors §

The host must provide data containment mechanisms to prevent consumption and further propagation of data in a poisoned TLP by a TVM or components in the TVM TCB.

It is strongly recommended that the host implement suitable protection schemes such as parity or ECC on its internal data buffers and caches to detect data integrity errors. If uncorrectable data integrity errors were detected, then the host must poison the data to prevent consumption and propagation by TVM, TVM assigned TDIs, or other components in the TVM TCB. The host must scrub registers that log information about the error, such as the syndrome, that could reveal confidential data.

11.5.7 TSM Tracking and Handling of Locked Root Port Configurations (Informative) §

The TSM must track attempts to modify registers or other changeable configuration controls affecting Root Ports connecting to devices in CONFIG_LOCKED or RUN. Precisely what must be tracked is implementation specific, and because the security properties of the device depend on correct implementation of these mechanisms, it is strongly recommended that persons skilled in building and validating secure hardware be deeply involved in the design and validation for all implementations. § Table 11-34 provides some general guidance regarding architecturally defined registers. Root Complex ~~↑implementation-specific↓~~ ↑implementation specific↑ registers must be evaluated to determine if modifications to those are allowable. Read-only registers, hardware initialized registers, and registers used as selectors for reading out data (e.g., the Power Budgeting Data Select register) are excluded from this table. The TSM must ensure that attempts to modify those registers cannot affect the security of associated TDI(s).

Table 11-34 Example TSM Tracking and Handling for Root Port Configurations §

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|---|---|--|
| <u>Cache Line Size</u> , <u>Latency Timer</u> , <u>Interrupt Line</u> | Allowed | |
| <u>Command Register</u> | See description | Clearing following bits causes the hosted IDE streams to transition to Insecure state: |

Base 6.4 vs Base 6.3

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|--|---|---|
| | | <ul style="list-style-type: none"> • <u>Memory Space Enable</u> • <u>Bus Master Enable</u> <p>Modification of other bits is allowed.</p> |
| <u>Status Register</u> | Allowed | |
| <u>BIST Register</u> , <u>Base Address Registers</u> , <u>Expansion ROM Base Address</u> , <u>Primary Bus Number</u> , <u>Secondary Bus Number</u> , <u>Subordinate Bus Number</u> , <u>Root Port Segment Number</u> | Error | Transitions hosted IDE streams to Insecure State. |
| <u>I/O base</u> , <u>I/O Limit</u> | Allowed | |
| <u>Secondary Status Register</u> | Allowed | |
| <u>Memory Base</u> , <u>Memory Limit</u> | Error | Transitions hosted IDE streams to Insecure State. |
| <u>64-bit Memory Base</u> , <u>64-bit Memory Limit</u> | Error | Transitions hosted IDE streams to Insecure State. |
| <u>I/O Base Upper 16 Bits</u> <u>I/O Limit Upper 16 Bits</u> | Allowed | |
| <u>Bridge Control Register</u> | Allowed | |
| <u>PCI Power Management Capability</u> | Allowed | If a power transition leads to the port losing its state, then the IDE streams hosted by that port transition to Insecure state. |
| <u>Device Control</u> | See description | Modifying <u>↑↓Extended Tag Field Enable</u> <u>↑↓Extended Tag Field Enable</u> must cause streams hosted by the port to transition to Insecure state |
| <u>Device Status</u> | Allowed | |
| <u>Link Control</u> | See description | Disabling or retraining the link leads to IDE streams configured in the Root Port to transition to Insecure state |
| <u>Link Status</u> , <u>Link Status 2</u> | Allowed | |
| <u>Slot Status</u> | Allowed | |
| <u>Root Control</u> | Allowed | |
| <u>Root Status</u> | Allowed | |
| <u>Device Control 2</u> | See description | Modifying state of <u>↑↓10-bit Tag Requester Enable</u> <u>↑↓10-bit Tag Requester Enable</u> causes streams hosted by the port to transition to Insecure state. Modification of other bits is allowed. |
| <u>Device Control 3</u> | See description | Modifying state of <u>↑↓14-bit Tag Requester Enable</u> <u>↑↓14-bit Tag Requester Enable</u> causes streams hosted by the port to transition to Insecure state. |

Base 6.4 vs Base 6.3

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|--|---|---|
| | | Modification of other bits is allowed. |
| <u>Link Control 2 , Link Control 3</u> | See description | Modifications to these registers are allowed. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |
| <u>MSI Capability , MSI-X Capability</u> | Allowed | |
| <u>Lane Margining at the Receiver</u> | See description | Modifications to these registers are allowed. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |
| <u>Secondary PCIe Extended Capability , Physical Layer 16.0 GT/s Extended Capability , Physical Layer 32.0 GT/s Extended Capability , Physical Layer 64.0 GT/s Extended Capability</u> | See description | Modifications to these registers are allowed. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |
| <u>ACS Extended Capability</u> | Allowed | |
| <u>Latency Tolerance Reporting Extended Capability</u> | Allowed | |
| <u>L1 PM Substates Extended Capability</u> | See description | Modifications to these registers are allowed. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |
| <u>Advanced Error Reporting Extended Capability</u> | Allowed | As specified in § Section 6.2.3.2.2 , error mask register settings control reporting of detected errors, but do not block error detection. |
| <u>Enhanced Allocation Capability</u> | Error | Transitions hosted IDE streams to Insecure State. |
| <u>Resizable BAR Extended Capability</u> | Error | Transitions hosted IDE streams to Insecure State. |
| <u>FRS Queueing Extended Capability</u> | Allowed | |
| <u>Flattening Port Bridge Extended Capability</u> | Error | Transitions hosted IDE streams to Insecure State. |
| <u>Virtual Channel Extended Capability</u> | Allowed – see description | Root Port enforces transaction ordering when TC/VC mapping is changed, or arbitration tables are updated. |
| <u>Vendor Specific Capability</u> | See description | To be analyzed by the vendor based on the security principles provided by TDISP. |
| <u>Vendor Specific Extended Capability</u> | See description | To be analyzed by the vendor based on the security principles provided by TDISP. |
| <u>Designated Vendor Specific Extended Capability</u> | See description | To be analyzed by the vendor based on the security principles provided by TDISP. |
| <u>RCRB Header Extended Capability</u> | Allowed | |

Base 6.4 vs Base 6.3

| Register / Capability / Extended Capability | Example Response to Register Modification | Description |
|--|---|--|
| Root Complex Internal Link Control | See description | Modifications to this register is allowed. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |
| Multicast Extended Capability | Error | Enabling multicast mechanism is not supported for TEE-I/O operation. Transitions hosted IDE streams to Insecure State. |
| Dynamic Power Allocation Extended Capability | Allowed | A Root Port must guard against the part being placed outside of its specification. If the Root Port cannot reliably operate within the power allocation through mechanisms like throttling, frequency control, etc. then the Root Port must transition hosted IDE streams to Insecure state. |
| TPH Requester Extended Capability | Allowed | |
| DPC Extended Capability | Allowed | |
| Precision Time Measurement Extended Capability | Allowed | |
| Native PCIe Enclosure Management Extended Capability | Allowed | |
| Alternate Protocol Extended Capability | Allowed | |
| System Firmware Intermediary Extended Capability | Allowed | |
| Protocol Multiplexing Extended Capability | Allowed | A Root Port that supports PMUX must not transmit or receive IDE TLPs using PMUX packets. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |
| Data Object Exchange Extended Capability | Allowed | |
| Integrity and Data Encryption Extended Capability | See description | Modifying the stream control register, selective IDE RID association registers, or selective IDE address association registers of streams that are bound to TVM assigned TDIs is an error, and the stream transitions to the Insecure state. |
| Flit Error Injection Extended Capability | See description | Modifications to this register is allowed. If the modifications lead to a link failure, the IDE streams configured in the port transition to Insecure state. |

11.5.8 IDE Extended Capability registers §

The host must enforce the integrity of IDE Extended Capability registers of the Root Ports used to configure streams used for TEE-I/O in the Root Port. Following mechanisms among others may be used to provide such integrity protection:

1. Restrict write access to such registers to the TSM or components in TSM TCB.
2. Transition the corresponding IDE stream to Insecure state if attempted to be modified by entities other than the TSM or components in TSM TCB.

TSM may provide an interface for the VMM to manage the lifecycle of IDE streams used for TEE-I/O. The VMM may use this TSM provided interface to configure, enable, disable, and reclaim (including non-graceful shutdown/reclaim) stream control registers. The details of such interfaces that may be provided by the TSM to the VMM are outside the scope of TDSP.

11.6 Overview of Threat Model and Mitigations §

This section provides a very brief overview. It is strongly recommended that thorough threat model analysis be conducted by competent security expert(s) for all implementations.

11.6.1 Interconnect Security §

The interconnect used to attach the device to the host needs to be secure against threats from physical attacks on the links. Adversaries are expected to have the ability to use lab equipment, interposers, custom devices, Switch firmware modifications, Switch routing table modifications, debug hooks in the Switches and Retimers, etc. to capture the data, re-order the data, or drop data transiting the links.

The adversary is expected to have the ability to perform reordering of transactions that are not legally allowed by the interconnect protocol.

The adversary may attempt to reprogram the encryption keys and replay protection counters associated with the link protection schemes to violate the confidentiality or integrity of the transactions on the link or to replay transactions on the link.

The adversary is expected to have the ability to craft custom devices or exploit vulnerabilities in authentic devices to attempt to spoof the identities used by the interconnect protocol (e.g., RID, PASID) to bypass access controls based on these identities.

These threats are addressed by use of IDE to secure the TLPs that carry TVM data.

The integrity protection on the TLP headers helps detect tampering of the identities such as RID and PASID. The use of selective IDE streams enables detection of attempts by the requester to use a RID outside the range defined by the RID association registers. The device and host secure the IDE keys and SPDM secure session keys from entities not in the TVM TCB. The device detects modifications to IDE configurations as errors and transitions all associated TDIs to ERROR state. If an SPDM session transitions to session termination phase, then all IDE streams that had keys established over that session transition to insecure state and all TDI that were transitioned to CONFIG_LOCKED state over that SPDM session transition to ERROR. The host may either prevent modifications to IDE configurations or treat them as errors and transition the IDE streams to Insecure state.

A TDI can be associated with an IDE stream only if the IDE stream was keyed using the same SPDM secure session as that used for the LOCK_INTERFACE_REQUEST. A TDI can be associated with a P2P IDE stream only if the P2P IDE stream was keyed using the same SPDM secure session as that used for the BIND_P2P_STREAM_REQUEST. An IDE stream transitioning to Insecure state moves the associated TDIs to ERROR.

The host should guard against the following additional threats, using [implementation-specific](#) [implementation specific](#) mechanisms, for security of IDE and TEE-I/O:

- Configurations used by the host to route addresses to Root Ports should be protected to prevent rerouting of transactions to unintended destinations:
 - TVM initiated transactions to device memory
 - P2P requests routed through the host
 - Device initiated transactions to host memory

- Reset of the host Root Ports and other logic blocks to place them in HwInit state
- Debug modes that affect the confidentiality and integrity of IDE keys, IVs, routing configurations, and other functions of the host that affect TEE-I/O security objectives.

11.6.2 Identity and Measurement Reporting §

The adversary is expected to have the ability to build custom devices that mimic a legitimate device but be otherwise maliciously crafted to compromise the confidentiality and/or integrity of the TVM confidential data provided to the device.

The adversary may have control of the version of device firmware that loads even in authentic devices and be able to exploit vulnerabilities that may exist in specific versions of the firmware.

The devices may support debug capabilities that the adversary can invoke to affect the confidentiality and/or integrity data in the device or functional operations of the device.

The adversary is expected to use physical accesses and/or access to local/remote debug interfaces among others to attempt to subvert the root of trust of the device including the services provided by the device root of trust for measurement, reporting, identity, authorization, or update.

The adversary may have controls that allow downgrading a device firmware after the TVM has verified the measurements and may use this ability to exploit vulnerabilities in the downgraded version.

The adversary may have control on the version of firmware/software components that are loaded in the host that are in the TVM trust boundary and be able to exploit vulnerabilities that may exist in specific versions of the firmware/software.

The adversary may have physical access to the host and/or access to local/remote debug interfaces among others to attempt to subvert the TVM root of trust in the host.

These threats are mitigated by use of [SPDM] for identity and measurement reporting.

The devices implement security mechanisms to protect the root of trust in the device. Devices implement secure mechanisms to provision the device root of trust. Devices implement secure root of trust for measurement and protect the integrity of the measurement registers. Devices protect the Root of Trust (RoT) and Root of Trust for Measurement (RTM) from debug modes. TSM provides device binding information to the TVM such that the TVM can answer the questions outlined in § Section 11.2.7 to determine if the device if it is accepting a device in a secure state into its TCB.

11.6.3 TDI Assignment and Detach §

Assigning a TDI to a TVM includes providing access to the MMIO resources of the TDI, and establishing access controls on DMA from the TDI to the TVM memory.

The adversary may be software outside the trust boundary of the TVM, including other TVMs, devices that are not trusted by the TVM, or an adversary using debug interfaces, etc. to influence the correctness and/or integrity of the MMIO resource assignment to a TVM and correctness and/or integrity of the DMA translation tables.

An adversary such as an untrusted VMM, or a TDI controlled by software outside the TVM trust boundary (including an unrelated/untrusted TVM), may attempt to maliciously access the MMIO resources assigned to a TVM to affect the confidentiality and/or integrity of the registers and MMIO mapped resources in the device.

An adversary such as an untrusted VMM may attempt to map the MMIO resources assigned to a TVM in an incorrect order and thereby attempt to trick a TVM to access a wrong set of registers and/or resources in the TDI.

The adversary may attempt to exploit properties like address decode priorities by assigning overlapping MMIO resources to two or more TDIs to redirect the transactions.

The adversary may attempt to reprogram a TDI that is being used by a TVM to affect the functioning of the TDI to influence the confidentiality and/or integrity of the transactions on the link or confidential data in the TDI. Examples of such reprogramming may include modifying the MMIO BAR of the device to attempt to drop transactions.

The adversary may attempt to asynchronously change the device state by issuing resets to the device to cause the device to drop established protections when a TVM is actively using the device.

The adversary may have the ability to launch a maliciously crafted TVM that collaborates with the adversary in violating the confidentiality and integrity of the TVM data.

These threats are mitigated by the use of TDISP.

TDISP provides the protocol and security requirements to lock TDI configurations using a LOCK_INTERFACE_REQUEST, obtain a report of the locked TDIs using a GET_DEVICE_INTERFACE_REPORT, securely enabling the memory space and DMA for TVM access using START_INTERFACE_REQUEST. A nonce generated by the device when the TDI is transitioned to CONFIG_LOCKED and verified on request to transition to RUN provide the property that all transitions through the TDISP state machine occur due to TDISP requests generated in the same SPDM secure session.

The DEVICE_INTERFACE_REPORT provides a trusted report of the TDI configurations, the list of MMIO resources associated with the TDI, and the order in which they must be mapped into the TVM address space. The TVM and TSM use the DEVICE_INTERFACE_REPORT to enforce that all MMIO resources of a TDI are assigned to the TVM to which the TDI is assigned and that the MMIO resources are mapped into the TVM address space in the expected order. § Section 11.5.1 specifies the requirements on the TSM to enforce address translation integrity. § Section 11.5.2 specifies the requirements on the TSM to enforce MMIO access control. The TSM through these access control mechanisms ensures that accesses with ~~↓↑the~~ T bit ~~↓↑and/or the XT bit~~ Set can be generated to a MMIO register only by the TVM that has been allocated those resources. The device in RUN must only allow a Request to access memory with IS_NON_TEE_MEM Clear when the Request's T bit is ~~↓↑Set, ↓↑Set and XT Mode is not supported or not enabled, or when the Request's both XT and T bits are Set and XT Mode is enabled.~~ If the device supports IDE capability, accesses targeting memory with IS_NON_TEE_MEM Clear must be accepted only on an IDE stream bound to the TDI.

ECN: Base 6.3 XT△↔

TSM uses host specific mechanisms to enforce DMA access control. The TSM uses the STOP_INTERFACE_REQUEST to ensure that all worked queued into a TDI has been drained and stopped before the resources allocated to a TVM can be reclaimed.

The ~~↓↑TSM~~ ~~↓↑TSM~~ is expected not to bind any P2P streams using BIND_P2P_STREAM_REQUEST message unless ~~↓↑all the following conditions are satisfied:~~

ECN: Base 6.3 XT△↔

- the host supports ATS translation, and
- ~~↓↑the XT Mode is enabled for the Requester device, or~~ the Root Complex performs the correct TEE access checks at the time an ATS Translation Request is issued. ~~↓↑Upon receipt of a Translation Completion that resolves to an address described by a P2P stream, the device receiving the Translation Completion is assured that it has full permission to request access to the specified resource. Similarly, any device that has a P2P stream configured and that receives a request on that stream with the T bit Set is assured that the sender has full permission to request access to the specified resource. No additional access checks are required or expected to be performed by the Receiver.~~

ECN: Base 6.3 XT△↔

TSM allows a TVM to update MMIO attributes of a TDI using SET_MMIO_ATTRIBUTE_REQUEST if the pages in the MMIO range of the request were allocated to the TVM making the request.

Devices track configurations of TDIs in CONFIG_LOCKED to detect attempts to reconfigure the TDI. Function level resets transition the TDI to ERROR. Conventional resets require the device to clear residual TVM secrets, IDE secrets, and SPDM session secrets such that they are not accessible to entities outside TVM trust boundary.

12. Architectural Out-of-Band Management §

12.1 Introduction §

“Sidebands” are physical interfaces other than the high-speed Lanes connecting two Ports, typically used for out-of-band platform management functionality. Out-of-band (OOB) management refers to the ability to communicate with peripheral subsystems, such as adapters and components, without interacting with firmware/software running on the host CPU(s).

This chapter provides a “toolkit” with a variety of potentially applicable technology ingredients, including, for example [SMBus] addressing, discrete signaling consolidation and Flexible I/O signal function negotiation, and how to utilize enhanced interfaces such as [I3C-Basic] and managed USB 2.0. The intent of centralizing this content in this document is to yield major improvements in capability and consistency to the physical, electrical, and logical domains for both systems and existing or new future form factor specifications or revisions.

The following table provides relative comparisons of typical architectural out-of-band interfaces and basic use guidance. *Bandwidth* is shown as the theoretical interface speed and is not indicative of practical throughput. The term *Isolated Control Plane* means that the interface does not involve the Link. For example, PCIe Vendor-Defined Messages (VDMs) are typically used for out-of-band management that does not involve firmware/software running on the host CPU(s) but does traverse Links. Some applications choose to isolate such traffic for reasons including host bandwidth or security.

Table 12-1 Relative Comparisons of Typical Architectural Out-of-Band Interfaces §

| General Out-of-Band Interface Usage Guidance | Transfer Rate | Isolated Control Plane | Example Applications |
|--|--------------------------|------------------------|---|
| Discrete/Physical Signals | Signal propagation delay | Y | PERST#, PRSNTNT#, PWRBRK#, PWRDIS |
| Serialized/Virtual Signals (e.g., PESTI) | =250 Kb/s | Y | Real-time virtual signals, OOB Controls/status |
| [<u>SMBus</u>] or [<u>I2C</u>] | 100, 400, 1000 Kb/s | Y | EEPROM, I/O Expanders, Sensors, MCTP, Security |
| [<u>I3C-Basic</u>] | 12.5 Mb/s SDR mode | Y | Hub Control, MCTP |
| USB 2.0 | 480 Mb/s | Y | Serial, Ethernet (NC-SI and TCP/IP), Mass Storage, MCTP, Security, Bridges to SPI, [<u>I2C</u>], and [<u>I3C</u>] |
| PCIe VDM | Link-specific | N | MCTP |

12.2 Framework for Sidebands §

Sideband signals refer to all signals that cross a defined interface, other than those associated with the high-speed Link itself. Sideband signals impact platforms such as through added cost, additional conductors and circuits for optional signals that might be rarely or never used. This negatively impacts cross compatibility and establishes legacy constraints that might impede more efficient solutions. Although sideband signals typically travel in parallel with the Link, sideband signals often interact with components other than the two that implement the Ports at each end of the Link.

The architectural framework defined here provides:

- A partial catalog of sideband signals already in common use for defined form factors.
- Reference physical topologies representing how sideband signals are used at the platform level.
- Extensible mechanisms for sideband capability discovery and control.
- Roles and responsibilities of various platform elements in relation to sideband signals.

The goal of the framework is to simplify the definition and evolution of form factors and platforms by providing form factor agnostic consistency and structure to sideband signal handling.

12.3 Sideband Signaling Mechanisms §

This section describes three methods of sideband handling for form factor and other interface specifications to use as tools. These include fixed function interfaces using discrete sidebands, Flexible I/O (Flex I/O) and Peripheral Sideband Tunneling Interfaces (PESTI). Each of these mechanisms are utilized where appropriate with the following sections providing specific guidance.

12.3.1 Discrete Sidebands §

Discrete sideband signals are physical wires and/or pins that have diverse functions such as those related to platform power up (e.g., **PERST#**), Link control (e.g., **CLKREQ#**), or non-Link functionality (e.g., 2-wire). These physical wires can be transmitted using different methods (e.g., copper, optical). Discrete sideband signals are not limited to digital, two-state or a single functional purpose or direction. Considerations for incorporating discrete sideband signals include architectural needs (e.g., **PERST#**), low latency needs, (e.g., **PWRBRK**), or fault domain minimization control (e.g., **PWRDIS**). Form factor specifications are permitted decide to incorporate such signal types when weighing functionality against interconnect size and cost.

In general, vendor specific platform requirements drive fan-in or fan-out requirements for each signal (e.g., whether **CLKREQ#** is independent for each function).

Examples of discrete signals incorporated in multiple form factors:

- **PERST#** - PCIe Reset (and similar component reset mechanisms) is a signal that causes a component to execute a Fundamental Reset. See § [Section 6.6.1](#) for details.
- **PRSN#** - Presence (and similar component presence mechanisms) is a class of signal that provides an indication of the presence of the adapter, component, or cable, and optionally the insertion or removal of an adapter, component or cable. See § [Section 6.7.1](#) for details.
- **PWRDIS** – Power Disable (and similar component power disable mechanisms) is a signal that causes an adapter or component to disable use of its own power. See § [Section 6.7.3.3](#) for details.
- **PWRBRK#** – Emergency Power Reduction (and similar adapter or component emergency power reduction mechanisms) is a signal that causes an adapter or component to reduce its own power consumption to a safe level. See § [Section 6.24](#) for details.
- **CLKREQ#** – Clock Request (and similar platform clock request mechanisms) is a signal that causes a platform to provide reference clock to the adapter or component. See § [Section 8.6.8](#) for details.
- **WAKE#** – Wake Up (and similar platform wakeup mechanisms) is a signal that causes a sleeping platform to restore main power to the adapter or component. See § [Section 5.3.3.2](#) for details.

An example of a signal with alternative solutions:

- **SMRST#** – SMBus Reset is a discrete sideband for recovering SMBus hang or stuck busy type conditions without interfering with the form factor's primary workload. This signal is a good candidate for exclusion or serialization, since it is high latency and may be rarely or ever used. It is common practice for applications that do not have a discrete **SMRST#** signal to use bus reset methods such as SMBus clock low reset or by executing SCL pulses followed by a STOP condition.

12.3.2 Flex I/O Sidebands §

Flexible input/output signals/interfaces (Flex I/O) utilize the concept where interface pins are permitted to be flexibly (re)purposed following discovery and determination of mutual capabilities and applying rules that guarantee safe and backward compatibility, where applicable, across the electrical and logical domains. When represented as a signal, form factor specifications should use the **FLEXIO-[N:0]** nomenclature.

Flex I/O interfaces provide the following benefits:

- Improved hardware interface utilization compared to fixed function pins.
- Opportunity to continually leverage evolving high-speed management interfaces.
- Flexibility to include, at present or future, value-added out-of-band functionality between platforms and peripheral subsystems, such as adapters or components.

IMPLEMENTATION NOTE: FLEX I/O PINS ARE NOT CALLED RESERVED §

Flex I/O signals are purposefully not called Reserved (RSVD) pins, which are typically specified to include no defined usage within a specific form factor specification version. RSVD pins typically call out both the platform and adapter sides to not connect any functional circuit, except high frequency filters such as on the CEM connector.

Flex I/O signal/interface engagement requires these steps, with guidelines described below:

1. Default State
2. Discovery
3. Compatibility Check
4. Control Negotiation

12.3.2.1 Flex I/O Default State Guidelines §

A Flex I/O interface pin is permitted to default to:

1. **Unused** : Unconnected (electrical open) or with a specific termination as dictated by a specific form factor or connector/cable specification.
2. **Pre-Wired/Inert** : This is where a pin/interface is wired with a predetermined circuit and connectivity but does not perform the intended functionality until after negotiation is complete. Voltage bias before negotiation is allowed. An example is a form factor specific required Flex I/O signal connected to platform logic that defaults to tristate/high impedance with a platform-side pullup resistor to the platform side's +3.3 V auxiliary power rail. Then after negotiation is complete, the signal is allowed to be used for its intended purpose. Form factor

specifications must ensure that any allowed default circuit conditions are electrically safe and do not impose logical domain constraints. An example is a cross power domain electrical stress or leakage condition.

3. **Switchable Function** : This is where a pin/interface defaults to a pre-defined function (often a legacy function), but then is electrically or logically switchable to an alternate function. An example is a form factor interface that defaults to JTAG functionality, with all the associated requirements such as default terminations and connectivity. Then following a positive discovery and compatibility check (defined below), a control disengages the default functionality and then engages the intended alternate functionality.

12.3.2.2 Flex I/O Discovery Phase Guidelines §

When used, Flex I/O capabilities are required to be described in FRU Information (See § Section 12.6). Form factor specified usages of Flex I/O signal(s) or interface(s) must be provided for discovery purposes via form factor specified FRU Information descriptors. Examples include, but are not limited to, pin number, pin functionality, supported power domain(s), bus topology and any signal integrity attributes deemed necessary to achieve the intended functionality. The [CEM-6.0] provides an example.

12.3.2.3 Flex I/O Compatibility Check Guidelines §

Failure to conform to the following guidance could result in undefined behaviors or effects not limited to electrical damage.

1. The platform must compare the adapter's advertised Flex I/O capabilities with the platform side interface capabilities and policy, if applicable, before attempting to perform any of these actions:
 - a. Change from a pin or circuit default state.
 - b. Communicate new control settings.
 - c. Utilize Flex I/O pin functionality in any way, such as toggling Flex I/O connected GPIO.
2. Mismatch: If a Flex I/O pin or interface functionality mismatch is detected between a platform capability or policy and interposer, adapter, or component, then the above three actions must be avoided.
3. Single-ended Flex I/O functionality on differentially optimized signals is allowed. Implementers must be aware of potential effects between such single ended signals, such as crosstalk being an unintended signal aggressor on a victim, impedance mismatches and other unintended effects along the signaling paths.
4. Differential interfaces utilizing single-ended optimized Flex I/O signals is prohibited due to differences, such as target impedance, improper GND shielding, board routing, connector design, etc.
5. Flex I/O usage and signal engagement may be applicable to one or more specific peripheral supported power states (such as Main power state only versus Auxiliary power and Main power states). If a peripheral advertises more than one supported power state, it is allowed to have a control command (if such a command is advertised as supported) for selecting a subset of power states that the peripheral must adhere in reference to specific Flex I/O interface(s). It is the responsibility of the peripheral to provide the necessary cross power domain isolation to avoid inadvertent behaviors.
6. Flex I/O usages are permitted to be independent of an adapter's or component's Link state.
7. Single-ended signal voltage negotiation for Flex I/O pins is outside the scope of this specification.
8. Using Flex I/O pins as power supply rails is prohibited, due to possible ill effects such as an incorrect or malicious software setting causing electrical stress or damage.

12.3.2.4 Flex I/O Control Negotiation Guidelines §

Failure to conform to the following guidance could result in undefined behaviors or effects not limited to electrical damage. See [11.5 Section 12.7](#) for the control/command protocol detail. Dynamic reconfiguration of Flex I/O is left to the implementer.

There are four scenarios for how Flex I/O controls assume their intended functionality:

1. Pre-wired circuit with configurable direction, purpose, signaling or protocol: This scenario occurs when an interface is pre-wired before any negotiated functionality. The interface must remain inactive until supported uses are discovered, and the requisite compatibility and policy check are complete, and the appropriate negotiations or controls are executed. Upon completing the negotiation, communication is permitted to commence from either side, as applicable.
2. Only platform side control is necessary: Following a successful discovery and compatibility check, engaging an interface might only require platform side control. For example, [CEM] supports legacy JTAG terminations by default. Upon discovery of a Flex I/O USB capability and compatible platform support, the platform must change from the default JTAG pin functions to USB 2.0. Note that this can be done with one control since no special break-before-make is needed to be electrically safe and USB 2.0 specification compliant.
3. Only peripheral or adapter side control is required: Commanding an adapter to perform specific engage or disengage operations must use a to-be-defined MCTP-based control method.
4. Both the platform and adapter side controls are necessary : Commanding a peripheral subsystem to perform specific engage and/or disengage operations must use a to-be-defined MCTP-based control method. Commanding the system side to perform specific engage and/or disengage operations is permitted to use implementation specific controls, such as a multiplexor select. It is important to use the engage and disengage order and other guidelines below.

12.3.2.5 General Flex I/O Control Guidelines §

1. Engagement Order: “Engaging” a Flex I/O is defined as enabling desired interface circuitry, connectivity and/or starting communication. Caution should be taken when electrically engaging an interface. Where applicable, the following order is advised.
 - **First** : Platform side. This is the PCI Express Link downstream facing port side.
 - **Second** : Interposer, adapter, or component side. This is the PCI Express Link upstream facing port side.
2. Disengagement Order: “Disengaging” a Flex I/O is defined as disabling or reverting to default interface circuitry, severing connectivity and/or ceasing communication. Where applicable, the following order is advised.
 - **First** : Interposer, adapter, or component side. This is the PCI Express Link upstream facing port side.
 - **Second** : Platform side. This is the PCI Express Link downstream facing port side.
3. Re-configuration Order: Reconfiguration might be desired between two functions or from an active Flex I/O interface type back to default state. The “break-before-make” paradigm is recommended by using the disengagement, followed by engagement order prescribed above. The duration between the break and the make steps is form factor specific. An example is to ensure that a higher voltage function is sufficiently drained before engaging a lower voltage function that might be intolerant to the higher voltage.
4. Forced Disengagement: Flex I/O functions are required to disengage upon the power state of the platform, interposer, adapter, or component dropping below the minimum negotiated power level or outside of the specified electrical operating conditions.

5. Flex I/O Run-time Reconfiguration: Run-time reconfiguration of Flex I/O is allowed when adhering to the guidelines stated within this specification. The logical domain handling of run-time change in Flex I/O functionality is implementation specific.
6. Reset Flex I/O back to defaults: If a Flex I/O signal changes functionality when main power is enabled, it must go back to its default state upon loss of the main power. If Flex I/O functionality changes when only auxiliary power is applied, it must go back to default upon loss of auxiliary power. Adapter or component use of non-volatile information to automatically revert Flex I/O to a previous state is prohibited.

12.3.3 Peripheral Sideband Tunnelling Interface (PESTI) Sidebands §

12.3.3.1 PESTI Introduction §

Peripheral Sideband Tunnelling Protocol (PESTI) is a single wire, half-duplex, bidirectional Universal Asynchronous Receive Transmit (UART) communication channel with a protocol that enables optional discovery and optional, real-time, virtual wire tunnelling between platform and peripheral agents. PESTI enables the minimization of the number of discrete sidebands signals, while extending functionality via scalable, form factor specific, extensible, and real-time virtual wires. PESTI is point-to-point with fanout support that utilizes a bus snooping and bus switching command method. PESTI provides message integrity detection support. PESTI protocol leverages common UART framing to create a simple, low logic, low cost out-of-band payload and signal tunnel. PESTI communication is based on a command and response method (no interrupt) where the peripheral/target is permitted to only transmit data upon request by the initiator. In some applications, PESTI protocol is permitted to occur on top of a traditionally static presence signal, thereby not incurring any extra signal cost for this interface.

This specification describes the base physical and protocol layers, from which form factor specifications are permitted to customize for specific attributes and virtual wire purposes. § Figure 12-11 shows a typical PESTI example for tunnelling virtual wired through a PCI Express Cable for managing the discrete sidebands on an interposer, adapter, or component.

PESTI is not intended to duplicate or replace what can be done over other non-real time interfaces. PESTI also does not currently support higher level protocols inclusive of message encryption.

NOTE: Open Compute Project Modular-PESTI §

The Open Compute Project also supports the identical “Modular-PESTI” [<https://www.opencompute.org/wiki/Server/DC-MHS>] interface and protocol.

Base 6.4 vs Base 6.3

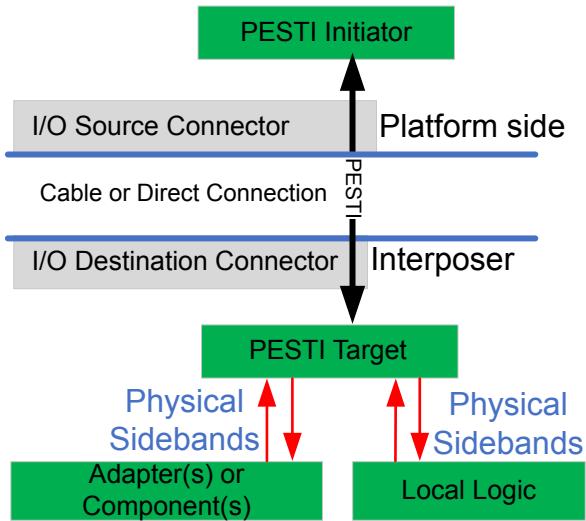


Figure 12-1 Example PESTI Application §

12.3.3.2 PESTI Physical Interface §

PESTI utilizes UART framing, operates directly at 3.3 V logic levels, and is a single, half-duplex signal:

- +3.3 Volt LVCMOS (open drain)
- 250,000 BAUD +/- 2%
- 8-O-1 (8 data bits, Odd Parity, 1 stop bit)
- Half-duplex

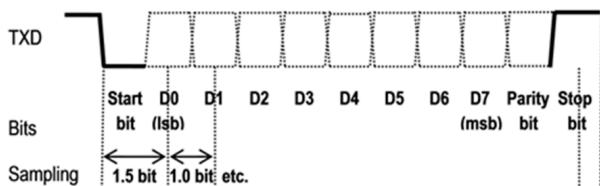


Figure 12-2 UART Data Framing §

12.3.3.3 PESTI Electrical Circuit §

The circuit shown in § Figure 12-3 is such that an unpowered ($V_{3P3_Target} = 0$ Volts) component holds the PESTI wire low indicating simple presence. Once the component has power and can respond to a command, the target General Purpose Input/Output (GPIO) sourcing the gate of the P-FET will be driven low. Once V_{3P3_Target} is engaged by turning the P-channel FET on, a rising edge of the PESTI signal is observed at the initiator. For this mechanism to operate as intended, the default state of the target GPIO must be an input (high impedance).

When used as simple presence, the initiator side circuit must remain the same as shown in § Figure 12-3 . The adapter or component side circuit can be simplified to a pulldown resistor with a maximum value of 2.27 Kohm +5% tolerance and minimum nominal value of 200 ohm -5% tolerance.

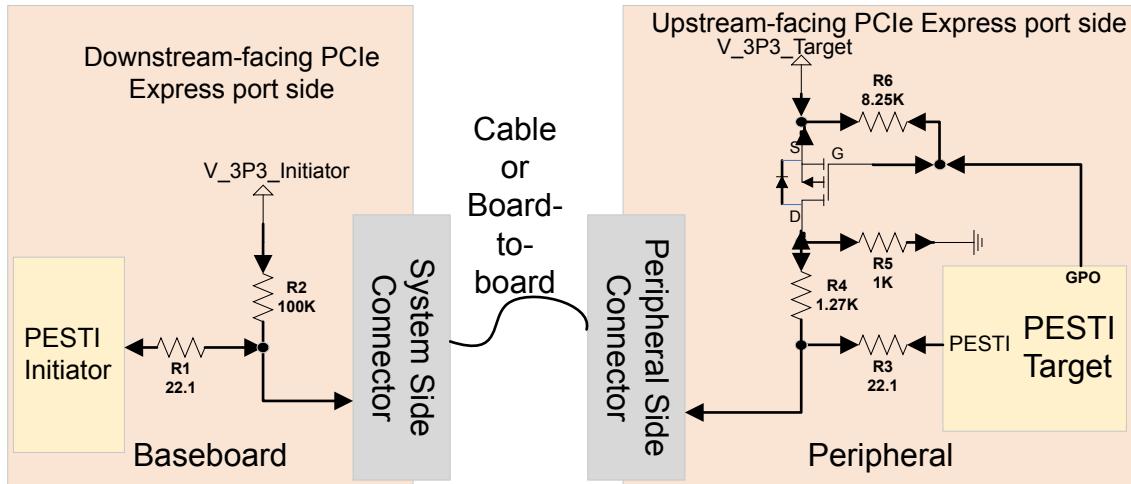


Figure 12-3 PESTI Circuit Diagram §

For non-hot-plug applications, PESTI is permitted to double as the presence signal, thereby incurring no added pin cost. Doing so for hot-plug applications is discouraged due to potential ambiguity or delay in determining a physical presence change versus an unresponsive target. PESTI is permitted to be utilized to indirectly tunnel physical presence status of hot-plug capable downstream targets.

IMPLEMENTATION NOTE: ELECTRICAL COMPONENT SELECTION FACTORS §

Referring to § Figure 12-3 above, **R1** and **R3** are recommended series termination resistor values. The values might require adjustment to the driver and transmission line characteristics.

R2 must be selected considering:

- Select the maximum value to minimize the current sourced into an unpowered target.
- Logic high=1 must be guaranteed to meet the baseboard initiator logic Vin High Minimum when cable/PESTI target is not attached/present.

R4/R5 must be selected considering:

- **R4** and **R5** (pull-down) and **R2** (pull-up) must guarantee a logic low=0 at the baseboard initiator logic if detection of an unpowered target is desired.
- **R4** value provides the rise time to a logic high at the initiator or target for an open-drain interface.
- **R5** provides a path to GND for the current sourced by **R2** on the source board to drain or bleed voltage accumulation at an unpowered target.

12.3.3.4 PESTI DC Specifications §

Table 12-2 PESTI DC Specifications §

| Symbol | Parameter | Min | Max | Units | Comments |
|--------|--------------------------|-------|-------|-------|--------------|
| VDD | Bus Voltage | 3.135 | 3.465 | V | 3.3 +/-5% |
| VIH | HIGH level input voltage | 2.0 | | V | 3.3 V LVCMOS |
| VIL | LOW level input voltage | | 0.8 | V | 3.3 V LVCMOS |
| VOL | LOW level output voltage | | 0.4 | V | |

12.3.3.5 PESTI Target Detection §

Presence of a PESTI target is characterized by a rising edge of PESTI. The default input state of PESTI at an initiator is HIGH=1 indicating that there is no adapter or component present. If the PESTI signal is observed as static low for more than t_{DBREAK} (See § Table 12-5), that target adapter or component is indicating simple presence. A rising edge of the PESTI signal indicates to the initiator that a target is ready to receive and process commands.

A UART BREAK event is when the wire is being driven/held low for a time greater than the entire frame length. Example: At 250 KBAUD, the PESTI frame length is nominally $4 \mu s * 11$ -bit positions which is $44 \mu s$. PLD logic is permitted to detect a BREAK condition by starting a timer at the falling edge of the signal. A valid BREAK assertion does not require a falling edge to be detected. In some cases, target UART receiver logic is permitted to not implement a BREAK detect status register. Depending upon the UART implementation, a BREAK event is permitted to present itself in status registers as an abnormal frame with a data value = 00h that has both a Parity Error (PE) and a Framing Error (FE: Stop bit = 0).

A valid PESTI frame is characterized by a Start bit of LOW=0, Stop bit of HIGH=1, and an odd number of ones (including Parity).

Target Detection Rules:

1. An initiator must not attempt communication while PESTI is held low (simple presence) by the target.
2. A PESTI target must not release BREAK until it is ready to respond to the payload request command.
3. Minimum low (BREAK) assertion width required to guarantee detection at the initiator is t_{DBREAK} .
4. An adapter or component is permitted to request a re-start of the discovery process by asserting and releasing BREAK at any time prior to the transition to the active phase (see § [Section 12.3.3.6.2](#)).

The PESTI circuit shown in § [Figure 12-3](#) guarantees that BREAK will be asserted from the time that platform input power is applied to the time that the target is powered and ready to respond to a command.

12.3.3.6 PESTI Protocol Commands §

12.3.3.6.1 Discovery Payload Request (DPR) §

PESTI compliant targets must support the following command.

- Discovery Payload Request: Data Byte Value = 00h

Following PESTI target presence detection, the initiator will transmit the payload request command to the target. The target response is to transmit the contents of its discovery payload for capture by the initiator. See ~~↑↓See~~ § [Table 12-6](#) for payload requirements. The command is permitted to be sent autonomously by a PLD or under the direction of Systems Management Firmware (SMFW). See § [Section 12.3.3.9](#) for initiator interface registers accessible to SMFW.

12.3.3.6.2 PESTI Virtual Wire Exchange (VWE) §

PESTI compliant targets must support the following command.

- Virtual Wire Exchange: Data Byte Value = 01h

Once a target's attributes (number, type, and order of virtual wires) have been identified by parsing the payload, an exchange of virtual wires between initiator and target is permitted to commence. This phase of the protocol is known as the active phase. Virtual wires are optional, and a target is not required to support any virtual wire bytes in or out following the VWE command.

If a PESTI target does not support virtual wires, it must respond with an acknowledge data byte (value = 00h) after receiving the VWE command.

The acknowledge feature can be utilized as a health monitor of the target. Alternatively, SMFW is permitted to disable the active phase by clearing APEN =0 (see § [Section 12.3.3.9](#)) to disable communication to a target after the discovery phase.

12.3.3.6.3 PESTI Fanout MUX Control §

The MUX switch control command must be supported in PESTI fanout control components only.

- MUX/switch control: Data Byte Value = 02h

See § Section 12.3.3.14 for additional details.

12.3.3.7 PESTI Initiator Abort §

A PESTI transaction sequence is defined as a series of data bytes exchanged between an initiator and a target. A sequence begins with the initiator transmitting one or more data bytes to the target. The sequence ends when the initiator receives the expected number of data bytes from the target. During the active phase, a target is required to recognize that a sequence is being terminated prior to its completion. The conditions for requiring termination of a sequence are architecture and implementation specific.

During Initiator Transmission:

In place of a valid data frame, the initiator is permitted to assert a BREAK condition to cancel a sequence that is in progress. The initiator must not interrupt a data frame after the Start bit has been sent. If an event occurs that requires an abort after a Start bit has been transmitted, the initiator must not assert BREAK until t_{TIDLE} (See § Table 12-5) after the previous Stop bit has been transmitted. When this sequence is observed by a target, it must ignore previous data bytes in that sequence and prepare for the start of the next sequence.

During Target Transmission:

The initiator abort assertion is permitted to occur at any point within a target transmitted frame. It is permitted to also occur during the active phase turn around period between initiator transmit and target transmit. In this case, it would be asserted following the Stop bit of the final byte transmitted by the initiator. The assertion width (t_{ABREAK} : See § Table 12-5) is guaranteed to be detected by a target that samples the PESTI logic state prior to transmitting any Start bit in the sequence. When an abort is detected, a target must not attempt to transmit any additional bytes and prepare itself for a new sequence start from the initiator.

12.3.3.8 PESTI Broadcast §

To minimize initiator logic, a single UART instance is permitted to be shared (multiplexed) among several targets. Multiplexing requires that targets be serviced in a round-robin rotation. Round-robin servicing has a limitation that responses cannot be captured from multiple targets simultaneously. However, an initiator is permitted to implement logic to transmit a command to several targets in parallel.

A broadcast sequence is characterized by the first byte of the sequence beginning with a data byte value of FFh. The second byte of the sequence is the command. An initiator must not use the broadcast command to request data or an acknowledge from any target. A target must not respond to the broadcast command with data or an acknowledge back to the initiator.

An example application is the Power Brake (e.g., PWRBRK#) sideband signal is permitted to be a virtual wire used to rapidly reduce power consumption of multiple adapters or components. PESTI initiators that implement a shared UART must support broadcast of a Power Brake event.

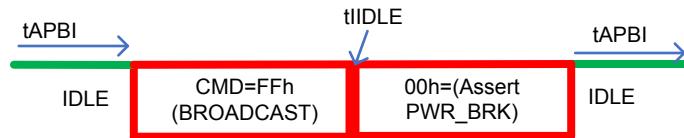


Figure 12-4 PESTI Broadcast Command §

Round-robin servicing naturally staggers PWR_BRK de-assertion during a virtual wire exchange which is a common platform preference to avoid excessive inrush current from high power loads. Additional delay between PWRBRK# de-assertion to multiple targets is implementation specific and controlled via SMFW and initiator logic that feeds into the appropriate PESTI channel.

An initiator is permitted to issue broadcast commands back-to-back without waiting for a response.

12.3.3.9 PESTI Initiator Control and Status Registers §

§ Table 12-3 defines the initiator register bits enable control and status indicators for each connected PESTI wire.

Table 12-3 PESTI Initiator Control and Status Registers §

| Field Name | Description | | | Attributes |
|---|-------------|---|--|------------|
| DISCOVERY_STATUS [1:0] (a.k.a. DSTAT) | 00b | No BREAK detected; PESTI wire is static HIGH=1 (<i>Default</i>) | | RO |

01b PESTI wire is static LOW=0 (simple presence).

10b Discovery payload has been received with a good checksum.

11b BREAK release detected, but payload has not been successfully received.

Initiator implementation requires the following behavior:

- Once a BREAK release is detected, the allowable DSTAT values must exclude 00b until the next fundamental PLD reset or cycle of platform input power.
- If APEN =1b and DSTAT =10b, then the target component is in the active phase and a BREAK condition on the PESTI wire must not transition DSTAT to a value of 01b. If APEN =0b, then a BREAK asserted by the target will result in DSTAT =01b. See § Section 12.3.3.13 for more information.

Table 12-4 PESTI Discovery Status State Transitions §

| DSTAT Current State | DSTAT Future State | State Input |
|---------------------------|--------------------------|--|
| xxb (Any State) | 00b (Absent) | Power on Reset (POR) |
| 00b (Absent) | 01b (PRSNT#) | BREAK asserted following POR de-assertion or an unpowered adapter or component is present or a static presence |

Base 6.4 vs Base 6.3

| Field Name | Description | | | Attributes |
|--|--|----------------------------|--|------------|
| | DSTAT Current State | DSTAT Future State | State Input | |
| | 01b (PRSNT#) | 11b (PESTI Presence) | BREAK release by the target, simple presence or PESTI component removal prior to target BREAK release while platform power is on. | |
| | 11b (PESTI Presence) | 10b (Payload Good) | Discovery Payload received without any errors | |
| | 10b (Payload Good) | 11b (PESTI Presence) | BREAK assertion and release by target if not in active phase. Payload Good is protected/locked during the active phase against target resets. This transition also occurs if DPEN is toggled from 0 to 1 while the component is still in the discovery phase (APEN =0.) | |
| DISCOVERY_PAYLOAD_ENABLE (a.k.a. DPEN) | 0b <i>(Default)</i> Initiator will not send payload request command. 1b Initiator will send payload request command, with retries, until DSTAT =10b. Initiator logic implementation requires that a transition of DPEN from LOW=0 to HIGH=1 by SMFW must clear any “Discovery Complete” flag indicated by DSTAT =10b. As a result, DSTAT would transition to 11b and the DPR command would be transmitted to the target. | | | RW |
| ACTIVE_PHASE_ENABLE (a.k.a. APEN) | 0b Initiator will not enter the virtual wire exchange phase for that PESTI instance. 1b <i>(Default)</i> Initiator will enter the active phase if DSTAT [1:0]=10b | | | RW |
| ACTIVE_PHASE_ERROR (a.k.a. APERR) | 0b <i>(Default)</i> Response RX error (i.e., parity, framing) or timeout has not occurred. 1b Response RX timeout or data error has occurred since last cleared (sticky). SMFW must write a 1b to this bit to clear the error once it has been acknowledged. | | | RW1C |
| ACTIVE_PHASE_STAT (a.k.a. ASTAT) | 0b <i>(Default)</i> Target component is in the discovery phase. 1b The Initiator and Target communication is in the active phase Completion of discovery is characterized by both of the following being true. 1. Discovery payload has been captured by the Initiator without any errors. 2. APEN =1 | | | RO |

Base 6.4 vs Base 6.3

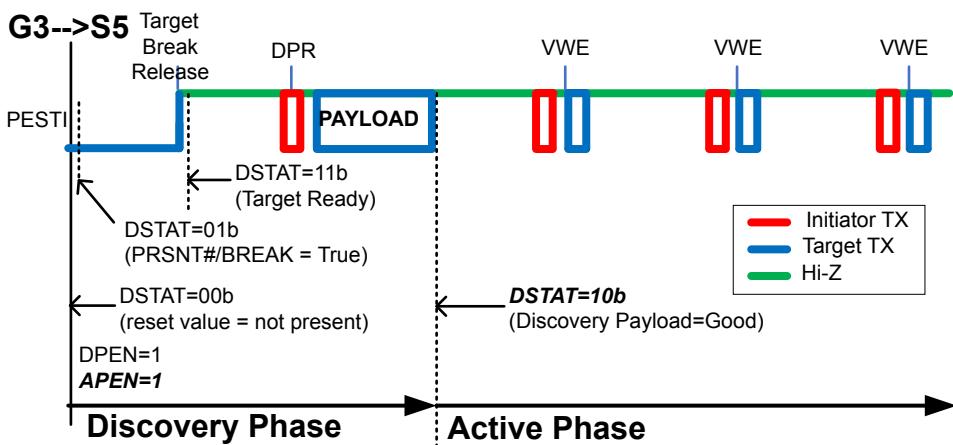


Figure 12-5 PESTI Protocol Phases §

12.3.3.10 PESTI AC Specifications §

Table 12-5 PESTI AC Specifications §

| Symbol | Parameter | Min | Max | Units | Comments |
|--------------|---------------------------------------|--------|--------|-------|--|
| t_{BAUD} | BAUD rate | 245000 | 255000 | Hz | 250000 +/- 2% |
| t_{FRAME} | Start +8b Data +Parity +Stop | 42.7 | 45.3 | μs | $1/t_{BAUD} * 11$ bits |
| t_F | Fall Time | - | 120 | ns | Same as 1 MHz [SMBus] (VIH,MIN + 0.15 V) to (VIL,MAX - 0.15 V) |
| t_R | Rise Time | - | 120 | ns | Same as 1 MHz [SMBus] (VIL,MAX - 0.15 V) to (VIH,MIN + 0.15 V) |
| t_{SPIKE} | Noise Spike suppression time | 0 | 50 | ns | Same as 400 kHz [SMBus]. Noise suppression is recommended. |
| t_{ABREAK} | Initiator Abort BREAK assertion time | 50 | 55 | μs | Synchronous to a normal START of frame when the initiator is transmitting. Asynchronous to START when the target is transmitting. |
| t_{DBREAK} | Target Discovery BREAK assertion time | 50 | - | μs | PESTI target BREAK can persist |
| t_{BPAUSE} | Broadcast Pause | 20 | - | μs | Pause to allow fanout snooper completion of channel enables |

Base 6.4 vs Base 6.3

| | | | | | |
|---------------|-------------------------------------|------|------|---------|---|
| t_{IIDL} | Initiator End of STOP to START | 0 | - | ns | STOP is typically sampled at the midpoint of the bit, so there is approximately 2 μ s of time to the next START. |
| t_{TIDL} | Target End of STOP to START | - | 1000 | ns | This is required for the target to sample for initiator abort prior to START of target TX. |
| t_{MARK} | End of BREAK to START time | 3.88 | 4.12 | μ s | MARK time is 1 BAUD period between end of initiator abort BREAK and START of new command |
| t_{DPTAR} | Discovery Phase Turnaround Time | 100 | - | ns | MAX not specified. It is bound by payload size and t_{DPRTO} . |
| t_{APTR} | Active Phase Turnaround Time | 0.1 | 200 | μ s | Between Target RX and Target TX of a response. MAX reduces time to sample initiator BREAK/Abort signal just before START. Target MCU should not have trouble meeting minimum time required to allow the initiator to prepare for RX following TX. |
| t_{DPRTO} | Discovery payload receive timeout | - | 250 | ms | Allows for 150 ms t_{DPTAR} + 2048-byte payload size. |
| t_{APRTO} | Active Phase receive timeout | - | 500 | μ s | Includes margin beyond t_{APTR} MAX + 8* t_{FRAME} MAX + 7* t_{TIDL} MAX = 389 μ s |
| t_{APBI} | Active phase bus IDLE time | 10 | - | μ s | Between initiator RX from target and initiator TX to same target |
| t_{MSC_WDT} | Mux Switch Control Watchdog Timeout | 100 | - | ms | Time for a snooper to detect a stuck bus following a downstream channel select operation |

12.3.3.11 PESTI Discovery Phase §

A PESTI target is required to support the DPR command when simple presence is not implemented. Payload format requirements enable the initiator implementation to be common and target agnostic. The discovery payload is typically consumed by the initiator logic (e.g., SIZE, CHECKSUM) and SMFW. The payload contents can be scaled to meet vendor requirements.

Discovery Phase Command and Target Response

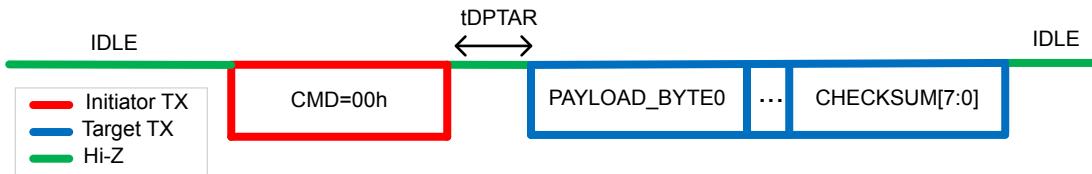


Figure 12-6 PESTI Discovery Command and Response Format §

Discovery Payload Turnaround Time (t_{DPTAR}): Maximum time allowed between initiator completing transmission of the DPR command and the response beginning to be received at the initiator.

Discovery Rules

Following the BREAK de-assertion by a PESTI target, the initiator must transmit the discovery request command to that target if DPEN =1.

- Turnaround time from target command received to start of response transmission has a minimum requirement of t_{DPTAR} .
- A target must complete the discovery payload response within the payload RX timeout of t_{DPRTO} .
- If a target BREAK condition is observed by the initiator with APEN =0, DSTAT [1:0] must be transitioned by logic to 11b (Discovery not complete) until a new payload is received successfully.
- The initiator must not transmit the VWE command unless a successful discovery payload is received and APEN =1.
 - Success means within the RX timeout period (t_{DPRTO}) with no byte parity or framing errors and a verified checksum.
- While DPEN =1, an initiator must continuously attempt to retrieve a discovery payload from a PESTI target that is present.
 - Example: Round-robin servicing by a single initiator UART to multiple targets:
If the discovery payload is not successfully received after an initial attempt plus two retries per target, the next target in the round-robin rotation will be serviced. When servicing returns to the target, the discovery request command is sent again as a set (initial + two retries) until successful.

If a target is unresponsive, the minimum time between successive transmissions of the DPR command from initiator to any target is t_{DPRTO} . Retries described in the example above are applicable to round-robin implementations only and not a requirement of this specification and is left to the initiator implementation.

Discovery Payload

The minimum required payload (8 bytes) is shown in § Table 12-6 for the case when simple presence is not implemented. The payload size must be a multiple of eight bytes including the checksum.

Table 12-6 PESTI Discovery Payload §

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----------------------------------|---|---|---|---|---|---|---|
| 00h | **PAYLOAD_FORMAT_VERSION [7:0]** | | | | | | | |

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------------|--|------|------|------|------|------|-----------------|----------------------|
| 01h | **DISCOVERY_PAYLOAD_SIZE [7:0] ** | | | | | | | |
| 02h | **VENDOR_ID [15:0] ** | | | | | | | |
| 03h | **VENDOR_ID [7:0] ** | | | | | | | |
| 04h | **DEVICE_CLASS [7:0] ** | | | | | | | |
| 05h | **DEVICE_ID [15:8] ** | | | | | | | |
| 06h | **DEVICE_ID [7:0] ** | | | | | | | |
| 07h | RSVD | RSVD | RSVD | RSVD | RSVD | RSVD | **PEC_SUPPORT** | **VENDOR_ID_FORMAT** |
| 08h to (N-1)h | Optional Vendor Specific Region (Varies)+ **Padding (0 – 7 Bytes)*** | | | | | | | |
| Nh | **Checksum** | | | | | | | |

Notes:

**Required region

***Minimum value of N=1, Maximum value of N=2040

PAYOUT_FORMAT_VERSION[7:0]

Indicates the payload format version. This field must be 02h.
01h is used by the Open Compute Project.

DISCOVERY_PAYLOAD_SIZE[7:0]

Indicates the total static payload size. The value of this bit field represents (payload total # of bytes / 8) - 1.

VENDOR_ID[15:0]

Utilize the PCI-SIG assigned Vendor ID, when available. If the vendor does not have a PCI-SIG Vendor ID (which applies to PESTI included subsystems such as a power converter board or a cable), then set VENDOR_ID_FORMAT = 1.

DEVICE_CLASS[7:0]

Indicates the device class of the PESTI target peripheral

Note that a given form factor is permitted to define specific module classes.

DEVICE_ID[15:0]

Vendor specificwith intent to represent a unique type of device. Not intended for unit specific information such as a serial number.

VENDOR_ID_FORMAT

0 = PCI -SIG assigned Vendor ID of the company delivering the adapter or component with the PESTI target

1 = Non-PCI-SIG assigned Vendor ID. e.g., Open Compute Project assigned.

PEC_SUPPORT

1 means that the target supports packet error byte checking (PEC) on command, write data, and read data returned to the PESTI initiator.

RSVD

Reserved for future use

Vendor Specific Region

Optional, implementation specific and is recommended to contain its own sub-payload version.

CHECKSUM[7:0]

CRC-8 with polynomial=0x07, Seed=0x00 is required in the discovery payload.

12.3.3.12 PESTI Active Phase §

Once a PESTI is transitioned to the active phase, the initiator autonomously exchanges hardware controlled virtual wires with each target component. The HW controlled virtual wire inputs and outputs are target DEVICE CLASS[15:0] specific. The number of bytes in and out and their respective usages are fixed per device class.

If the number of virtual wire bytes out is zero, the initiator will skip transmitting Virtual Wire Output (VWOUT_0) and wait to receive any Virtual Wire Input (VWIN_N) bytes. If the number of virtual wire bytes is zero, then an acknowledge target response of 00h is required from the target, whenever one is expected at the initiator. SMFW is permitted to clear APEN =0 so that the VWE command is not transmitted, and the bus will remain idle until the next power on reset or re-entry to the discovery phase.

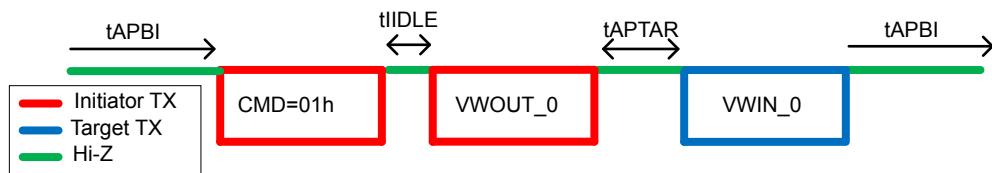
Virtual Wire Exchange Example (1 Byte Out/In)

Figure 12-7 Single Byte PESTI Virtual Wire Exchange §

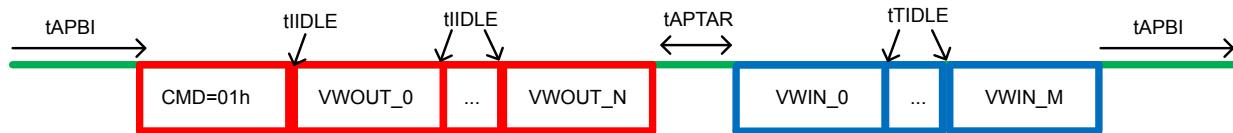
Virtual Wire Exchange Example (N Byte Out/ M Byte In)

Figure 12-8 Multi-byte PESTI Virtual Wire Exchange §

Active Phase Rules

1. Minimum PESTI idle period between initiator RX of the previous target response & TX of the next target command is tAPBI
2. While ASTAT =1, DSTAT must remain 10b (Discovery Complete)
3. A target assertion of BREAK while ASTAT =1 must result in APERR =1.

4. A target must wait a minimum of t_{APTAR} before transmitting a VWE response.
5. A target must complete a response to the initiator within the RX timeout of t_{APRTO} .
6. The maximum period between commands sent from an initiator to that same target is not bound by this specification. The active phase of any component is under the control of SMFW and can be disabled.
7. If target supports virtual wire OUTs only, the target must respond with 00h back to the initiator as confirmation of receiving the OUTs.
8. SMFW is permitted to clear APEN to disable the active phase for a component.
9. Asymmetrical number of out and in bytes is permitted.

The method to route virtual wires to/from internal platform logic or other entities and the PESTI is outside the scope of this document. The usage of the virtual wires (internal commands/policies or connections to local physical signals) by the target is also outside the scope of this document.

Example VWIRE Bit Mapping Definition

Below is an example of how a form factor, such as an interposer to a [CEM] slot, is permitted to map legacy wires onto virtual wires over PESTI.

Table 12-7 Example VWIRE_OUT_0 (Initiator to Target) §

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|----------------|------|---|------|------|------|--------|---------|--|
| RSVD | RSVD | RSVD | RSVD | RSVD | RSVD | S0_RUN | PWR_BRK | |
| PWR_BRK | | Active high (1=Assert, 0=De-assert) virtual wire | | | | | | |
| S0_RUN | | Active high (1=True, 0=False) virtual wire indicating platform power state is in ACPI S0. | | | | | | |
| RSVD | | Reserved for Future Use | | | | | | |

Table 12-8 Example VWIRE_IN_0 (Target to Initiator) §

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|-------------|------|--|------|------|------|------|------|--|
| RSVD | RSVD | RSVD | RSVD | RSVD | RSVD | RSVD | WAKE | |
| WAKE | | Active high (1=Assert, 0=De-assert) virtual wire indicating the target component request to enter the ACPI S0_RUN state. | | | | | | |
| RSVD | | Reserved for Future Use | | | | | | |

Packet Error Checking

Packet Error Checking (PEC) is optional in the initiator and target. PEC is recommended in the active phase when information integrity is critical to the application (e.g., an errant control may cause more ill effects than a bit flip on an analog read that gets averaged out over time). A target's PEC support is described in a discovery payload bit. PEC utilizes CRC-8. Enabling PEC adds to active phase latency.

When supported by the target and set by the initiator, the one-byte command (CMD) bit 7 indicates to the target that the PEC byte is being delivered following with the command byte.

Example: CMD=81h, VWOUT_0, PEC, Turnaround, VWIN_0, PEC

If the target indicates no PEC support in the discovery payload, then the initiator must not set bit 7 in the CMD nor deliver an extra PEC byte following the command. This is because following the turnaround, the target may drive contention on against the incoming PEC byte.

If the CMD to target has bit 7 set indicating PEC:

1. The initiator must supply the PEC byte.
2. If the PEC check at the target fails, then the target must drop the command and, if applicable, receive payload. A method to indicate from target to initiator that a PEC failure occurred is out of scope of this specification.
3. The target must append a PEC byte to the end of the active phase payload response (just as it does for the discovery payload).
4. If the initiator calculates a bad PEC on the target response message, the initiator must drop the response. Further handling, such as retries, is implementation specific.

For commands such as CMD=00h for get discovery payload, PEC is optional, since the 1-byte command has parity.

Example Supported Commands with and without PEC

01h = Put 1 Virtual Wire Byte, Get 1 Virtual Wire Byte

81h = Put 1 Virtual Wire Byte with PEC, Get 1 Virtual Wire Byte with PEC

12.3.3.13 PESTI Target Reset and Fault Handling §

If the target resets during the discovery phase prior to the entry to the active phase (e.g., APEN =0), the DSTAT value would revert to 11b (Payload not received.) The initiator would autonomously send the payload request command if DPEN =1. The target component would not be transitioned to the active phase until DSTAT =10b. If discovery had not previously completed or DPEN =0, the component would not be discovered and would not transition to the active phase.

If the target resets during the active phase, it must be observed as temporary unresponsiveness at the initiator. This would be reflected in the ACTIVE_PHASE_ERROR (APERR) register bit if any RX timeout or parity error occurred.

Following target reset, a BREAK assertion and release would be observed by the initiator. During the active phase (DSTAT [1:0]=10b & APEN =1), DSTAT must be locked. Locking the discovery status and payload data is required so that FW can safely consume the contents at any warm or cold reset. Since discovery has already occurred, the initiator would resume sending the virtual wire exchange command and the target would not observe a discovery request command. Once a component has transitioned to the active phase, the only method to unlock the discovery payload and status is by SMFW clearing, then setting DPEN .

12.3.3.14 PESTI Fan-Out §

Applications where the ability to fan-out a PESTI bus to multiple targets exist. Since the number N is not a priori known by the motherboard, and pre-plumbing for a maximum quantity requires additional interconnects, PESTI fan-out support helps scale the ability to support PESTI type features.

This specification version includes the scope of fanout from between 2 to 8 downstream buses. Although out of scope, nesting of multiple tiers of fan-out within a single hierarchy is possible with additional command codes and circuitry. In all fanout cases, it is left to the designer to understand the latency effects of fanout width (and depth). Two methods are shown where the MUX method is for typical fanout needs and the Switch method is targeted for applications that require the ability to broadcast commands simultaneously with all targets.

Two fanout methods of operation include the MUX and Switch methods (see circuits below):

- MUX method: Typical 1-to-many fanout. Broadcast commands are not supported.
- Switch method: For applications requiring broadcast commands to all attached targets.

Two modes of operation for a fanout controller include Target Mode and Snoop Mode:

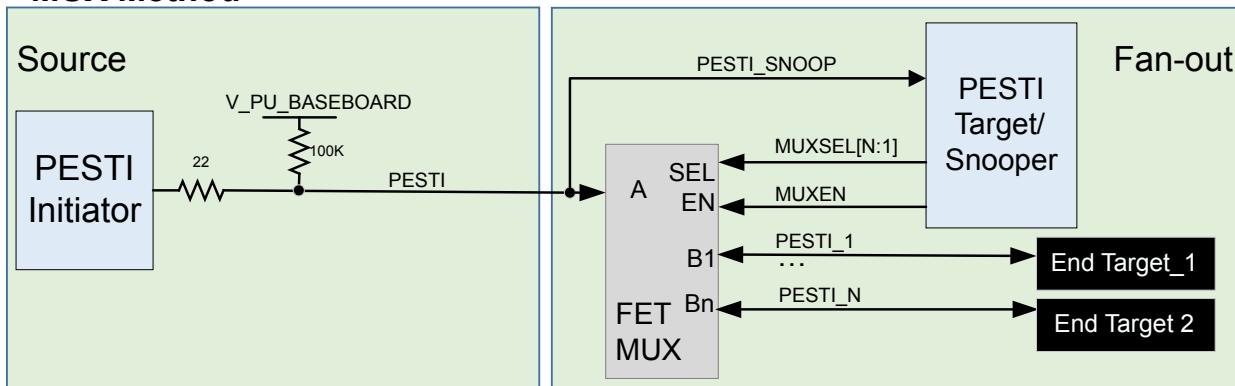
- Target Mode (Default):
 - Controller acts as a target supporting discovery and active phases.
 - Only the fanout controller is permitted to be attached to the initiator bus.
 - MUX Method: No channels are selected.
 - Switch Method: Fanout controller on Ch0 is always enabled; all others default disabled.
 - Must support =1 status command for the initiator reading information from the fanout controlling PESTI target such as:
 - The current MUX/switch settings
 - If an issue was observed such as a closed switch bus hang watchdog timeout
 - Live status of downstream subsegments (when the MUX or switch is open for a particular subsegment)
- Snoop Mode:
 - Fanout controller must enter snoop mode any time any other target(s) are attached to a bus.
 - Is permitted to process broadcast commands if application relevant.
 - Listen/Process only special fanout control commands (MUX Select or Switch channel enable), thereby ignoring discovery and target mode active phase commands intended for other targets.

Example MUX Select commands are permitted to include:

- Go to Target Mode and de-select all subsegments.
- Select specific subsegment(s) as connected, which can be comprised of 1, all or select groups.

Base 6.4 vs Base 6.3

MUX Method



Switch Method (for applications requiring Broadcast)

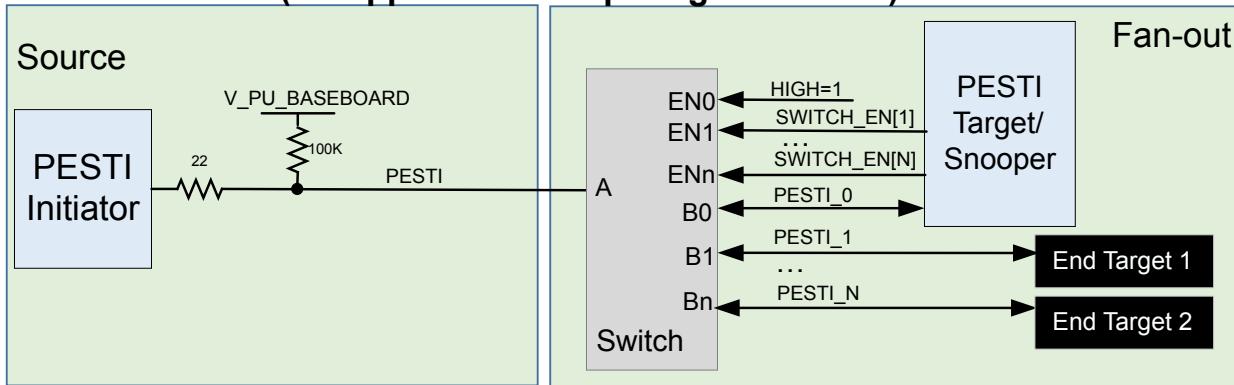


Figure 12-9 PESTI Fan-out Methods §

Buffer(s) might be necessary if the Target/Snooper and the FET MUX or Switch have significant layout stubs. The designer is permitted to choose a switch with active buffers versus a FET switch to avoid reflection effects.

Since any downstream interface is permitted to have a strap (such as a strong presence pulldown resistor) or be in a stuck state driving high or low, it is imperative to not create a deadlock condition where the Fanout controller cannot observe new commands to change the MUX or switch settings. Therefore, it is required to have the fanout controller implement a watchdog timer of at least t_{MSC_WDT} after closing one or more channels to oversee any signaling activity. If no such activity (rising or falling edges), the PESTI snooper ↑↓ must open ↓ ↑ must open ↑ the selected channel(s) after the timeout.

An alternative option is for the fanout controller to always be able to observe and report upstream the state of any downstream sub-channel so that the PESTI initiator can choose to not close a suspect channel.

MUX Switch Control (MSC) Command Structure

Table 12-9 MSC_CTRL_VAL (Initiator to PESTI Snooper Target) Data Byte Value = 02h §

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|---------------|----------------|---|--------|---|
| RSVD | RSVD | RSVD | DS_ALL_CH_SEL | DS_CH_SEL[2:0] | | ENABLE | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------------|---|---|---|--|---|---|---|---|
| ENABLE | | | | Active high (1=Enable, 0=Disable) When the mux or switch is disabled, only the snooper target can receive communication from the initiator. All downstream MUX or switch channels must be disabled. | | | | |
| DS_CH_SEL[2:0] | | | | Hex representation of downstream channel select (0...7 decimal). This bitfield is only used if ENABLE = 1b and DS_ALL_CH_SEL = 0b. | | | | |
| DS_ALL_CH_SEL | | | | When set to 1b, then all downstream channels are connected. Used for broadcast commands only. This bitfield is only valid if ENABLE = 1b. When this bitfield is set to 1b then DS_CH_SEL bitfield must be set to 000b. | | | | |
| RSVD | | | | Reserved for Future Use. | | | | |

Table 12-10 MSC_STAT_VAL (PESTI Snooper Target to Initiator) §

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---|---|---|------|---|---|---|--------------------------|
| | | | | RSVD | | | | NOT_ACK |
| NOT_ACK | | | | | | | | |
| 0b | | | | | | | | Success |
| 1b | | | | | | | | Error |
| RSVD | | | | | | | | Reserved for Future Use. |

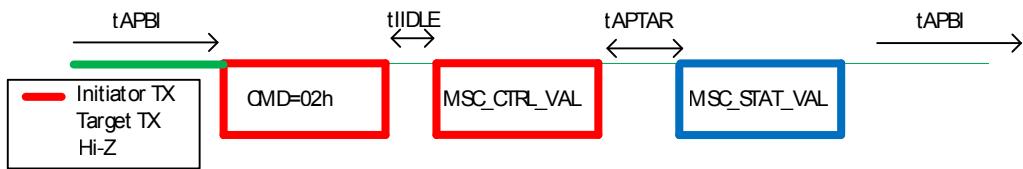


Figure 12-10 PESTI Mux Switch Control Format §

Fanout Rules:

1. PESTI snoopers must not be nested. Only a single fanout level is supported.
2. A PESTI snooper fanout target must default to all downstream PESTI channels disabled during the discovery phase.
3. A PESTI snooper must not respond to a discovery payload request command = 00h when any downstream channel is selected.
4. A PESTI snooper must not respond to any virtual wire exchange using command 01h when any downstream channel is selected.
5. Except for when broadcast commands are supported and transmitted, only a single downstream channel of a Switch can be selected at any given time.
6. Non-snooper targets must ignore argument(s) following the MSCMSC command = 02h.

7. On any PESTI that a fanout switch is present, the initiator must begin any broadcast with the following sequence:
- Broadcast start value = FFh
 - Wait for t_{BPAUSE} (Broadcast Pause)
 - Resume with typical broadcast sequence of FFh (repeated) followed by the broadcasted command value(s).
8. A fanout switch snooper must enable all downstream channels at any start of a broadcast indicated by 0xFF as the first, or repeated, data byte of a sequence.
9. Once the broadcast sequence is complete, the initiator must utilize the MSC command to select a single downstream channel of any switch or MUX prior to resuming typical communication.

12.3.3.15 PESTI Security Considerations §

Although PESTI is a basic, low-level data communication channel the following threats are identified with possible mitigations, although the specific mitigations are implementation specific.

Threats

1. Physical implant/signal re-routing: Assurance, where applicable, that the PESTI target is on the same HW (or same target) as other management interfaces. Security Protocols and Data Model (SPDM) types of security capabilities are the likely current method used to address this threat.
2. Physical implant, man-in-the-middle snooping or alteration of payloads or virtual wires in flight. Potentially mitigated with encrypted payloads most likely using SPDM defined methods.

12.4 Managed USB 2.0 §

Numerous use cases exist for USB 2.0 as a plug-and-play, out of band management interface, such as:

- Adapter or component firmware update, telemetry, debug, and security operations such as attestation, recovery, and measurement.
- MCTP direct to an adapter or component.
- TCP/IP or Network Controller-Sideband Interface (NC-SI) traffic.
- Bridge to common low-level interfaces such as GPIOs, UARTs, SPI, I3C and JTAG.

USB Rules:

1. The USB Host must be on the platform side of the form factor interface.
2. USB 2.0 high speed and full speed modes are supported.
3. USB 2.0 low speed and USB 1.1 mode devices are prohibited when directly connected to the form factor interface but are allowed when attached to a downstream port of a device side USB hub.
4. USB 2.0 low speed is prohibited due to USB 2.0's support of larger packet sizes as well as common handling of Flex I/O default terminations and D+ pullup-based attach detection (e.g., CEM JTAG vs USB Flex I/O alternate function rules).
5. USB 2.0 voltage through form factor interfaces must be 3.3 V to support standard discrete components such as bridges and hubs and to provide maximum signaling margin for long ~~busses.~~ buses. Voltage translation when connecting USB 2.0 to components that do not support 3.3 V is implementation specific.

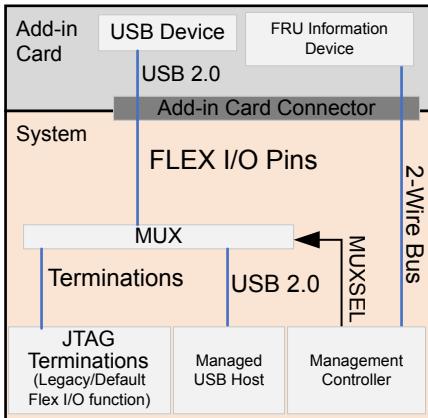


Figure 12-11 [CEM] form factor example circuit for repurposing legacy JTAG to USB 2.0 mode §

12.5 2-Wire Interface §

[I2C] and [SMBus] are long standing and useful bus interfaces that are expected to be fully supported for the foreseeable future. Form factor specifications dictate the electrical particulars, such as the reference voltage, power domain, peripheral capacitive loading and define such an interface as being required, optional or unsupported. Although the SMBUS specification is stable in terms of revisions, especially at the electrical level, it is always encouraged that form factor specifications attempt to reference the latest versions of the [SMBus] specification when possible. [SMBus-3.2] version 3.2 is the preferred minimum version at the time of this writing due to support for the Default Target Address feature, while using inclusive terminology. The remainder of this specification refers to [SMBus] and not [I2C]. However, [I2C] compliant components that are not [SMBus] compliant, such as an EEPROM or MUX, are permitted at the designer's discretion.

MIPI [I3C-Basic] is an additional protocol option that shares the same physical **2-wire** interface as the [SMBus] interface. This specification defines a backward-compatible method of discovering and enabling the optional [I3C-Basic] mode on the existing **2-wire** interface. The [I3C-Basic] architecture includes an optional intermediary component known as **Hub** (see § Section 12.5.3.3) to address specific [SMBus] challenges related to asynchronous communications. [I3C-Basic] defines the protocol, including its electrical characteristics. The recommended PCI Architecture to allow interoperability with legacy adapters is for the platform to use the **Hub** which provides a method for a transition between [I2C] / [SMBus] and I3C. Legacy [SMBus] and [I3C-Basic] have different electrical requirements which need to be considered in the platform design. The [I3C-Basic] provides a common electrical requirement definition across all PCIe form-factor connectors (interfaces).

Common Interface and Signal Names

Historically, the [SMBus] interface and signal names were used in various form factor specifications. With the addition of optional I3C interface on the same pins, the following generic terms should be utilized, when referring to either [SMBus] or I3C:

1. “2-wire interface”
2. Signal names “SCL” for the serial clock wire and “SDA” for the serial data wire

12.5.1 2-Wire Interface Use Cases §

Throughout this section, the generic term **↑↓2-wire↑↑2-Wire↑** interface references either [SMBus], [I2C], or [I3C-Basic].

Table 12-11 ↑↓2-wire↑↑2-Wire↑ Interface Example Usages §

| Functionality | Typical Applications |
|--|--|
| Field Replaceable Unit (FRU) Information | Discovery of Hardware inventory, capabilities such as bus topologies, features supported, etc. |
| Security | Attestation of HW/FW, Root of Trust for Measurement, Root of Trust for Update, Root of Trust for Recovery, Data Encryption |
| Configuration / Controls | Flex I/O, Link subdivision |
| Update | Peripheral Subsystem Firmware |
| Health | Error Causes, Crash Dump Logs, Temperature, External Link States |
| Telemetry | Power, Throughput |

12.5.2 2-Wire Addressing §

To ease topology and component discovery operations, § Table 12-12 lists common SMBus default target addresses recommended to be used by all PCI Express-based form factors, when such component exist. Components supporting both SMBus and I3C modes must stop responding to their original SMBus address(es) upon transitioning to I3C mode, as described in § Section 12.5.4.2 .

§ Table 12-12 identifies the recommended addresses for specific SMBus usages. Note that a card carrier is defined as one with a connector edge based on a form factor that includes connector slots based on one or more instances of the same or different form factor, sometimes referred to as passenger slots. An example is a [CEM] card carrier containing one or more M.2 form factor slots.

IMPLEMENTATION NOTE: RECOMMENDED 2-WIRE INTERFACE ADDRESSES §

All implementations are recommended to use the addresses in § Table 12-12 for the listed usages to ease discovery. For all usages, any address other than the ones listed in § Table 12-12 are permitted.

Platform makers should avoid these listed addresses on the same subsegments.

For maximum backward compatibility, platform makers should not include any component addresses on the same (sub)-segment that connects to the form factor.

The addresses shown have synergy with other industry standards such as [NVM-Express].

Table 12-12 Baseline SMBus Recommended Default Target Addresses §

| Default Target Address: Binary with x indicating R/W (Hex 8-bit/Hex 7-bit) | Recommended Usages |
|---|---|
| 1010 010xb (A4h/52h) | Recommended for FRU Information Device on a card carrier ¹ |
| 1010 011xb (A6h/53h) | Recommended for FRU Information Device on a PCI Express-based adapter or component ¹ . |
| 0011 101xb (3Ah/1Dh) | Recommended for Primary MCTP-compliant Management Target ² |
| 1101 010xb (D4h/6Ah) | Recommended for Secondary MCTP-compliant Management Target ³ |
| 1110 100xb (E8h/74h) | Recommended for Addressable SMBus MUX, if applicable |
| 1110 000xb (E0h/70h) | Recommended for ↓↑2-wire↓ ↑↑2-Wire↑ Hub ⁴ |

Notes:

1. The FRU Information Device is typically an optional element. Form factor specified features that need out-of-band discovery descriptors must make the FRU Information Device required and available in the appropriate earliest discoverable power state (auxiliary and/or main power).
2. Primary MCTP target: This target must operate when main power is applied. This interface is permitted to operate when only auxiliary power is applied.
3. Secondary MCTP target: This target is permitted to operate when auxiliary and/or main power is applied.
4. ↓↑2-wire↓ ↑↑2-Wire↑ Hubs must default to the static address listed and are then permitted to be changed via register programming or via [I3C-Basic] Dynamic Addressing.

Interposer discovery details are left to the implementer, with designer care needed to avoid address conflicts if the ↓↑2-wire↓ ↑↑2-Wire↑ interface extends into standard form factors.

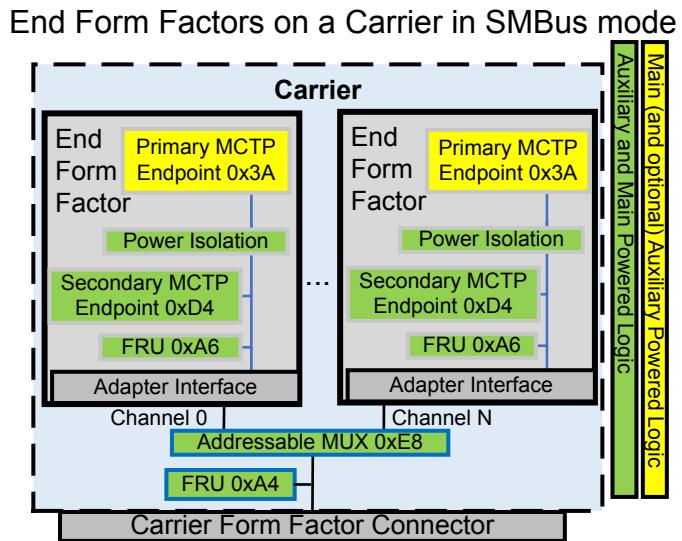


Figure 12-12 Example of 2-wire, 8-bit addressing for a card carrier with N end form factors in SMBus mode §

12.5.3 ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ Bus Sharing §

The subsections describe ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ interface sharing options for fanout on an adapter or component or between form factor instances.

12.5.3.1 ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ Multi-Drop Topology §

A multi-drop topology with a ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ interface is permitted. In platform topologies utilizing a multi-drop ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ interface, unique addresses must be assigned:

- Either statically (e.g., using a unique slot identifier control method) or
- Dynamically using SMBus Address Resolution Protocol (ARP) or [I3C-Basic]'s Dynamic Addressing.

Caution should be taken as the use of multi-drop ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ topology exposes the potential for malicious adapters or components to initiate traffic to other adapters or components on the same subsegment.

12.5.3.2 SMBus MUX Use §

An addressable SMBus MUX is defined as a bidirectional fan-out multiplexer with one upstream channel and one or more downstream channels configured by a ~~↑↓2-wire↓~~ ~~↑↑2-Wire↑~~ interface command (not discrete mux select signals) from a Management Controller to select zero or exactly one downstream channel to the upstream channel.

The use of addressable switches, which allow for connecting multiple downstream subsegments to the upstream segment at one time is not recommended, for reasons that include signal integrity, security and addressing complexities.

IMPLEMENTATION NOTE: SECURITY CONCERNS WITH SMBUS MUXES §

Platform implementations utilizing ~~↑↓2-wire~~ ↑↓2-Wire switches/MUXes are recommended to mitigate against security risks of errant or malicious adapters or components that could initiate peer-to-peer traffic to other adapters, components, or other platform adapters or components on the same ~~↑↓2-wire~~ ↑↓2-Wire segment.

Platform designers are recommended to consider that switches/MUXes interrupt the continuity of the shared bus breaking the arbitration and acknowledgment mechanisms of SMBus. This creates major challenges for implementations utilizing multi-initiator protocols, such as MCTP, especially when:

- MCTP messages initiated by downstream components are spread over long periods of time (e.g., as per [PLDM-Firmware-Update]), or
- asynchronous communication happens (e.g., asynchronous event or alert reporting as [PLDM-Platform-Monitoring-Control]), or
- response times may be difficult to predict, or response messages delayed (e.g., due to complex computations, such as cryptographic ones required for the SPDM attestation flows defined in [SPDM]).

Additional challenges in the above scenarios exist when switches/MUXes interrupt the continuity of the shared bus and:

1. the platform uses MCTP bridging (management controller is not the source of the MCTP communication and thus has more difficulty to predict the MCTP request/response patterns) or
2. downstream port enabling/disabling is not synchronized with SMBus transactions (e.g., truncating the MCTP packets).

Platforms utilizing MCTP protocol in such scenarios are recommended to use a ~~↑↓2-wire~~ ↑↓2-Wire Hub component instead (see the next section). Switching to [I3C-Basic] mode may be another alternative to address such challenges but only possible if all downstream components support [I3C-Basic] (I3C is a single-initiator protocol even when running MCTP and it also allows explicit control of I3C in-band interrupts).

12.5.3.3 ~~↑↓2-wire~~ ↑↓2-Wire Hub Use §

It may be beneficial to use a ~~↑↓2-wire~~ ↑↓2-Wire Hub component for reasons including:

- a. Bus capacitance isolation.
- b. Fanout of I3C and SMBus components with I3C in-band interrupt aggregation across all downstream channels.
- c. Voltage translation and power domain isolation.
- d. Using offload capabilities such as SMBus agents to send and receive asynchronous SMBus messages, such as those utilized by MCTP.

An example use of such a ~~↑↓2-wire~~ ↑↓2-Wire Hub is presented in § Figure 12-13 . This architecture is recommended to address the interoperability challenges existing with simple MUX components, as described in § Section 12.5.3.2 . SMBus endpoints can initiate asynchronous messages even if they do not support I3C mode of operation. I3C mode, which is typically available with such ~~↑↓2-wire~~ ↑↓2-Wire Hubs, allows protocol and voltage transition to enable coexistence of SMBus and I3C components.

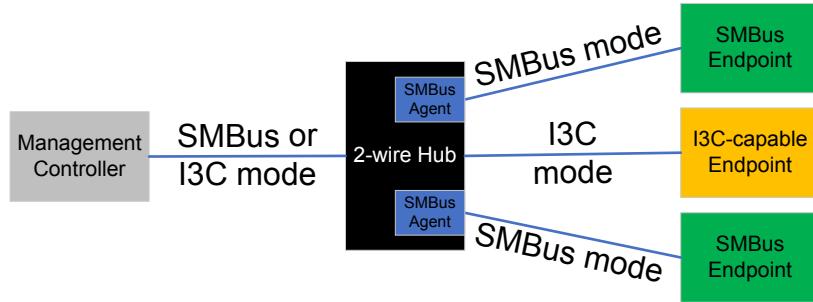


Figure 12-13 Example of ~~↓↑2-wire↓~~ ↑↑2-Wire↑ Hub Use §

~~↓↑2-wire↓~~ ↑↑2-Wire↑ Hub components should be discovered using the I3C DCR value of 0xC2. See the allocation at [I3C-DCR].

It should be recognized by implementers and form factor specifications that the high signaling rates drive for subsegment capacitance that is significantly lower than the SMBUS specification. This can limit the achievable bus speed or require utilizing intermediary logic such as I3C hubs when considering entire channel characteristics.

12.5.4 [I3C-Basic] Support on Existing SMBus Signals §

Most PCI Express-based platforms have supported [SMBus] as an optional ~~↓↑2-wire↓~~ ↑↑2-Wire↑ interface (i.e., Serial Clock and Serial Data) for adapter or component management purposes. This section defines [I3C-Basic] as an additional option where the [I3C-Basic] interface shares the same 2 wires as the SMBus interface.

While [SMBus] was defined with different electrical requirements for different form-factors, [I3C-Basic] is defined with one interoperable set of behaviors for PCI Express. This section defines the behaviors that:

- Determine which interface is active
- Define the electrical requirements for PCI Express [I3C-Basic]

12.5.4.1 I3C Basic Overview §

Key attributes of [I3C-Basic] support on existing [SMBus] signals are the following:

- Supports the MIPI™ Alliance Specification for I3C Basic, Version 1.0 available at [I3C-Basic] (See Implementation Note Optional I3C Features).
- Sideband ~~↓↑2-wire↓~~ ↑↑2-Wire↑ bus reliant use cases should be defined using either [SMBus] only or both [SMBus] and [I3C-Basic].
- If optional I3C Basic is supported, then I3C Data shares the same signal as SMBus Data, and I3C Clock share the same signal as SMBus Clock.
- If optional I3C Basic is supported, SMBus voltage levels must also be tolerated for backwards compatibility.
- I3C operating voltage must utilize 1.8 V signaling as defined in [I3C-Basic]. The rationale is for maximum bus length and allowed capacitance which eases implementations in large platforms.

IMPLEMENTATION NOTE: OPTIONAL I3C FEATURES §

The following I3C features from [I3C-Basic-1.1.1] should be supported if I3C is supported:

- The Target Reset pattern (RSTACT) with Common Command Code 0x2A (Broadcast) and 0x9A (Direct).
 - This allows the host to perform a I3C peripheral reset without having to go back to SMBus mode.
- Asynchronous Timing Control (Mode 0) (support determined if GETXTIME CCC is acknowledged)
 - This aids in scenarios needing real time communication by providing a timestamp.
- Grouped addressing (support determined through GETCAPS CCC)
 - This allows broadcast request to multiple I3C Basic components using a single message.
- SETBUSCON CCC
 - [I3C-Basic] recommends using SETBUSCON to inform I3C Targets of the specific use of the Bus, and that vendor specific CCCs will be used. Can also inform I3C Targets of the version of the specification that the Controller supports.

IMPLEMENTATION NOTE: POTENTIAL ELECTRICAL CONTENTION §

Due to the nature of [I3C-Basic] utilizing push-pull active drivers, implementers should be cautious about possible corner case scenarios where electrical contention could occur (e.g., an I3C component is actively driving data high while the initiator goes through a reset and drives data open drain low as part of a new command). Actions such as the initiator beginning a recovery sequence with an SMBus clock low timeout to reset the target is advised.

12.5.4.2 I3C Basic Discovery and Mode Changing §

The goal of the I3C Basic adapter or component discovery flow as shown in § Figure 12-14 is to enable an I3C Basic capable host and I3C Basic capable components to establish I3C Basic communication while allowing backward compatibility with legacy [SMBus] components. If one side of communication supports both [SMBus] and I3C Basic and the other side supports SMBus only, then SMBus protocol and voltage is used. If there is a mix of I3C Basic and SMBus components that are active on the same bus, then only [SMBus] protocol and voltage is used. Mixed Fast Mode is prohibited. This flow should be initiated after any power rail state change.

The discovery flow uses reserved address 1111 110xb (FCh 8-bit/7Eh 7-bit) to determine if there are components that support I3C Basic on the bus. Address 7Eh (7-bit) is reserved in the [SMBus] specification and cannot be assigned to any SMBus component. Address 7Eh (7-bit) is defined for I3C Basic and every I3C Basic component will respond as per the I3C Basic Specification. Any time a component that supports both SMBus and I3C Basic receives a cycle for address 7Eh (7-bit), it should disable SMBus (not drive or stretch the clock, nor ACK SMBus addresses) until reset to help the host determine if there are SMBus only components on the bus. If only I3C Basic components are detected and the host chooses to use I3C Basic, then the component requires a transition time from 3.3 V to the I3C Basic voltage within the time $T_{smb2i3c}$ (see § Table 12-14). A reset by driving a system management hardware mechanism (e.g., a hardware reset or power removal) if supported by a given formfactor or driving the clock low for a specified period reverts the interface

to SMBus at 3.3 V signaling within the time $T_{i3c2smb}$ (see § [Table 12-14](#)). An I3C Basic Target Reset issued by the host resets the I3C Basic interface but does not impact the signaling voltage. See § [Figure 12-14](#) for more details of this flow.

Base 6.4 vs Base 6.3

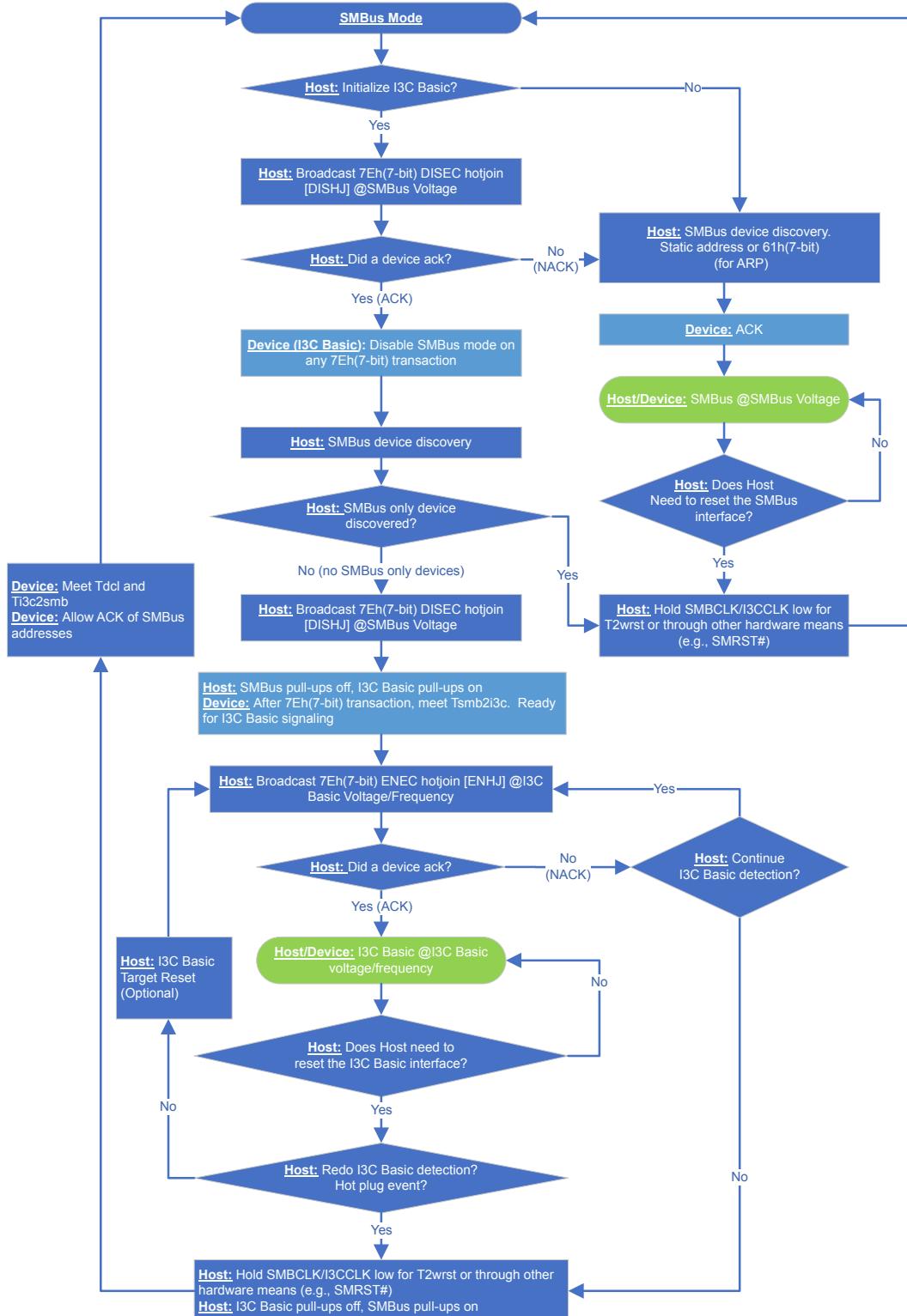


Figure 12-14 SMBus to I₂C Transition Flow §

12.5.4.3 I3C Basic DC and AC Signal Requirements §

The I3C Basic operating voltage and capacitance for SCL and SDA is defined in § Table 12-13 .

For more information on logic levels or bus timings, refer to [I3C-Basic].

Table 12-13 I3C Basic Logic Signaling DC Specification §

| Symbol | Parameter | Min | Nominal | Max | Unit | Notes |
|-------------|---|------|---------|------|------|-------|
| V_{ddi3c} | I3C Basic Operating Voltage | 1.65 | 1.80 | 1.95 | V | |
| C_{i3c} | Total adapter and component capacitance for I3C support | | | 20 | pF | 1 |

Notes:

1. Total capacitance the platform board will see from the adapter including all connector, board routing, component package wires, and component on die parasitic effects when the target component(s) is/are in I3C Basic mode. The capacitance of SMBus components on the I3C Basic interface even if not operational is also included.

§ Table 12-14 defines the component maximum transition times between [SMBus] and I3C. During $T_{smb2i3c}$, the host will transition the pull-ups from Vddsmb to V_{ddi3c} and the component will transition to V_{ddi3c} based signaling. During $T_{i3c2smb}$, the host will transition the pull-ups from V_{ddi3c} to Vddsmb and the component will transition to Vddsmb based signaling.

Table 12-14 I3C Timing Requirements §

| Symbol | Parameter | Min | Max | Unit | Notes |
|---------------|---|-----|-----|------|-------|
| $T_{smb2i3c}$ | Component transition time from SMBus to I3C Basic | | 20 | ms | 1, 2 |
| T_{dcl} | Component Clock low reset time | 25 | 35 | ms | 2, 3 |
| $T_{i3c2smb}$ | Component transition time from I3C Basic to SMBus | | 20 | ms | 2, 3 |
| T_{2wrst} | Host clock low timeout | 50 | | ms | |

Notes:

1. The host must wait a minimum of $T_{smb2i3c}$ before sending I3C commands.
2. No SMBus or I3C transmissions must occur during $T_{smb2i3c}$, T_{dcl} , and $T_{i3c2smb}$.
3. The host must wait a minimum of $T_{dcl} + T_{i3c2smb}$ before sending SMBus commands.

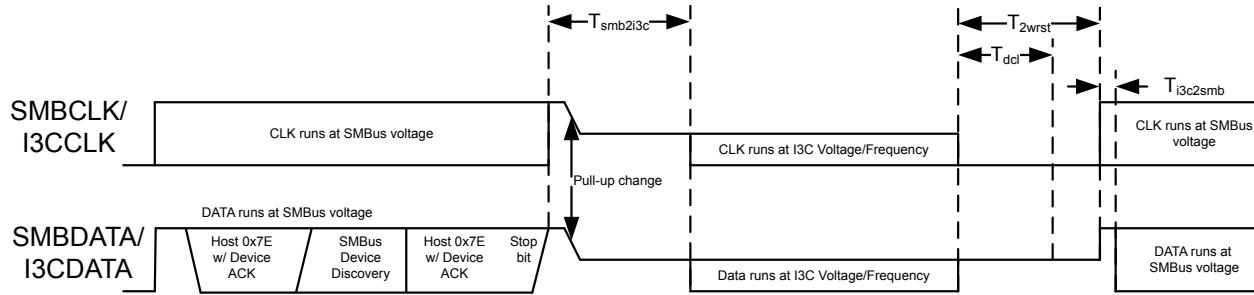


Figure 12-15 Component Timing Diagram for Transition to I3C Signaling Voltage §

Base 6.4 vs Base 6.3

IMPLEMENTATION NOTE: RATIONALE FOR T2WRST §

MIPI [I3C] reset mechanisms provide useful granularity. That specification's sections titled “Controller Stuck SDA Handling” and “Controller Recovery After a Crash or Unexpected Reset” describe various typically infrequent situational, diagnosis, and recovery methods. These methods provide numerous stimuli with various responses, inclusive of reducing drive strength to limit effects of possible electrical contention via a target actively driving SDA bus high or low.

Since:

1. The recovery methods are cumbersome.
2. A crash or unexpected reset of a controller may result in a lack of awareness of any target device's current mode of operation.
3. Any chance of causing electrical contention is not preferred.

Then, holding SCL low for the T_{2wrst} minimum duration (which may start at any point in a transaction) is the additional required recovery method, which makes the solution:

1. Easy to implement in software/firmware.
2. Easy to observe and debug on a logical bus trace.
3. Removes all concerns over the necessary electrical tuning and/or selection of controller and/or devices with specific drive strength granularity.

IMPLEMENTATION NOTE: DEVICE HOT PLUG WITH MULTI-DROP 2-WIRE BUS CONSIDERATIONS §

This specification explicitly prohibits running in mixed mode, which means at least one device runs in SMBus mode while other(s) run in I3C mode. If a system implements a multi-drop ~~↓↑2-wire~~ ↑↓2-Wire bus between multiple hot pluggable devices, then:

- Hot removal of a target is no issue, where discovery is outside the scope of this specification.
- An end form factor target that is inserted, internally reset or power cycled is required to start in SMBus mode with the respective form factor native voltage. If a multi-drop bus exists and is in I3C mode at the time of new device insertion, reset or power cycle, then the I3C mode 1.8 V voltage does not cause stress to the new target. Unexpected bus behavior may occur if the newly added, reset or power cycled device(s) apply internal SMBus pullup resistors or do not support the requisite glitch filter to ignore I3C traffic. How the system detects new device insertion and knows to send T_{2wrst} to force all targets back to SMBus mode is outside the scope of this specification.

12.6 Field Replacement Unit (FRU) Information §

FRU Information is data that describes an adapter or component. FRU Information includes data about the adapter or component such as:

- Inventory information (e.g., serial number, model number, etc.).
- Capability information (e.g., power consumption, performance, etc.).
- Information required by the host to configure the adapter or component (e.g., supported connector subdivision combinations, clocking modes supported, etc.).

FRU Information may be made available to a Baseboard Management Controller or service processor using the following mechanisms:

- Over the ~~↓↑2-wire~~ ↑↓2-Wire interface with use of the protocol defined in § [Section 12.6.1](#).
- Over ~~↓↑2-wire~~ ↑↓2-Wire interface, USB, or PCIe VDMs with use of the MCTP protocol.

In-band access of FRU Information via a method such as Management Message Passthrough (MMPT) or Memory-Mapped BMC Interface (MMBI) [MMBI] is outside the scope of this chapter.

12.6.1 FRU Information Device Requirements §

FRU Information is contained in a FRU Information Device. The following list contains requirements and recommendations for FRU Information Devices regardless of the operating mode being [SMBus]/[I2C] or [I3C]:

- A ~~↓↑2-wire~~ ↑↓2-Wire discoverable Field Replaceable Unit (FRU) Information Device must be implemented as a required element when dictated by form-factor specifications such as to support specific features.

2. The FRU Information Device is permitted to be any implementation that meets the requirements in this specification (e.g., a physical integrated circuit or virtually emulated by firmware).
3. The FRU Information Device is recommended to be protected from write operations that may corrupt or replace FRU Information. The exception is when the FRU Information is accessed via MCTP or while in a manufacturing mode. An adapter or component is permitted to update the FRU Information in conjunction with a firmware update.
4. The FRU Information Device must support at least 8 full writes.
5. The FRU Information Device must be accessible in any of the following power states:
 - a. Main power only.
 - b. Main power and auxiliary power if auxiliary power is supported.
 - c. Auxiliary power only if auxiliary power is supported.
6. If auxiliary power is supplied, then the availability and transitions of main power must not disrupt FRU Information Device access.
7. The FRU Information Device must be accessible on the ~~↓↑2-wire~~ ↑↓2-Wire interface within 1 second after power becomes valid and must remain accessible while power is valid unless otherwise specified.
8. Clock-low timeout-based reset recovery must be supported per the [SMBus] Specification or [I3C-Basic] section of this specification. The FRU Information Device must be accessible on the ~~↓↑2-wire~~ ↑↓2-Wire bus within 1 second after the de-assertion of clock-low timeout-based reset. It is recommended to reset the current address pointer due to clock-low reset.

12.6.1.1 FRU Information Device Requirements Specific to SMBus/I2C Mode §

This section describes the FRU Information Device requirements and recommendations when using SMBus/I2C read and write transactions.

1. The FRU Information Device must use a 7-bit target address.
2. FRU Information Device capacity must be an even power of 2 in the range of 256 Bytes up to 64 KB.
3. The FRU Information Device must support at least 400 kHz operation in SMBus/I2C mode. The adapter or component should limit bus capacitance. The appropriately tuned pullup resistors are the platform's responsibility. The adapter or component should have no or weak pullup resistors only.
4. FRU Information Device accesses must use 2-byte addresses for read or write offsets. One-byte offsets supplied by the platform during random reads should be zero extended to two-byte offsets. See § Figure 12-17 .
5. The FRU Information Device must maintain a current address pointer. The current address pointer must auto increment to the next byte after each byte that is read. Refer to the “Current Address Read” and “Current Sequential Read” in § Figure 12-17 . A read of the last byte of the last memory page of the FRU Information Device must cause the current read pointer to roll over to the first byte of the first memory page of the FRU Information Device. For example, for a 256-byte FRU Information Device, a read of the last byte of the FRU Information Device at offset FFh causes the current address pointer to roll over to 00h. The current address pointer must be cleared to 0h upon a power cycle of the FRU Information Device.
6. Any controller on the adapter or component must not initiate traffic to the FRU Information Device on the same adapter or component unless the FRU Information Device is hidden from host view (e.g., by detaching the FRU Information Device from the system-facing ~~↓↑2-wire~~ ↑↓2-Wire interface). For example, the FRU Information Device must be hidden from host view during a firmware update that includes new FRU Information. While the FRU Information Device is disconnected from the form-factor connector, accesses to the FRU Information Device via the ~~↓↑2-wire~~ ↑↓2-Wire bus must be Not Acknowledged.

7. If the FRU Information over the **$\downarrow\downarrow 2\text{-wire} \uparrow\uparrow 2\text{-Wire}$** interface is initially provided using a discrete SMBus/I2C FRU Information Device (e.g., a physical EEPROM), the component must be detached from the **$\downarrow\downarrow 2\text{-wire} \uparrow\uparrow 2\text{-Wire}$** interface when entering I3C mode. If the FRU Information Device is being emulated by a programmable component, such as a microcontroller, then when switching to I3C mode, the virtual FRU Information Device SMBus address must be logically disabled. As mixed fast mode is not supported, designs must not assume that the I3C bus frequency and short SCL high time allows for SMBus/I2C devices with 50 ns glitch filters to coexist.
8. Placing the FRU Information Device behind a **$\downarrow\downarrow 2\text{-wire} \uparrow\uparrow 2\text{-Wire}$** MUX or hub is prohibited due to possible variability in discovery method.

12.6.1.2 [SMBus]/[I2C] Access Protocol §

The following figures describe the proper logical sequence of supported FRU Information Device read and write commands when in SMBus/I2C mode. Note that the address example is specific to 4 KB component.

Base 6.4 vs Base 6.3

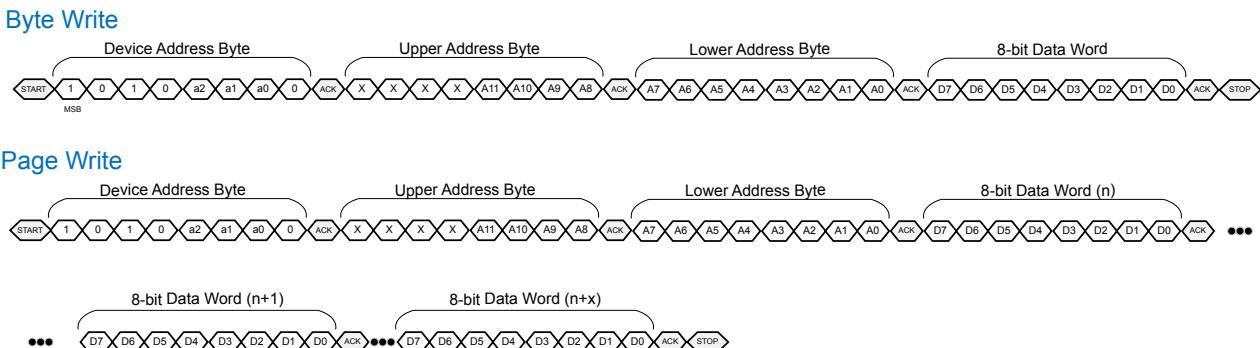


Figure 12-16 SMBus/I2C-based FRU Information Device Writes with Two-Byte Addressing §

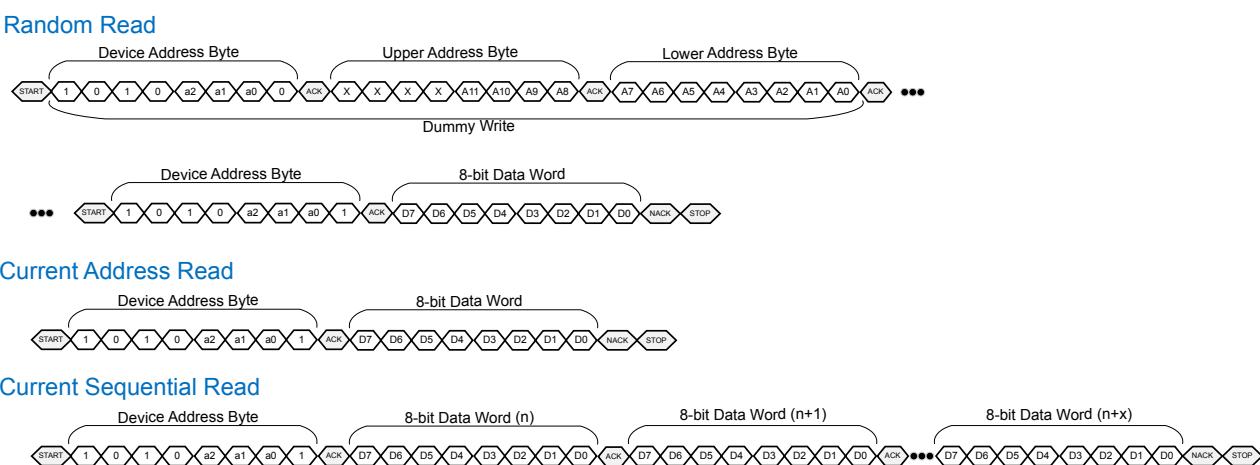


Figure 12-17 FRU Information Device Reads with Two-Byte Addressing §

12.6.2 FRU Information Format §

This section defines the format of the FRU Information stored in the FRU Information Device. The FRU Information format must comply with the requirements defined in the IPMI Platform Management FRU Information Storage Definition ([IPMI-FRU]).

In addition to CRC-based data integrity, assurance of data authenticity is permitted to support an optional cryptographic signature of the FRU Information Device. It is recommended that the FRU Information be covered by a digital signature of the adapter's or component's firmware and validated when the firmware is updated. If the content is validated during adapter or component firmware initialization, care must be taken to ensure the component and information are available to system management firmware as needed, such as when only auxiliary power is supplied.

The IPMI Platform Management FRU Information Storage Definition ([IPMI-FRU]) allows more than 1 MultiRecord to be in a single FRU Information Device. The follow table describes the specific MultiRecord format that applies to all PCI Express adapters or components.

Table 12-15 PCI-SIG MultiRecord §

| Bytes | Factory Default | Description | | | | | | |
|-----------------------------------|---|--|------|------------|---|---|-----|---|
| PCI-SIG MultiRecord Header | | | | | | | | |
| 0h | 0C0h | PCI-SIG Record Type ID: This field indicates the IPMI Record Type ID of the PCI-SIG MultiRecord. This field must be set to a value of C0h. | | | | | | |
| 1h | 2h or 82h | Record Format: This field indicates the format of this MultiRecord. <table border="1" style="margin-top: 5px;"> <tr> <td>Bits</td><td>Definition</td></tr> <tr> <td>7</td><td>When Set, this bit indicates that this is the last MultiRecord in the list.</td></tr> <tr> <td>6:0</td><td>This field indicates the MultiRecord format version. This field must be set to a value of 2h.</td></tr> </table> | Bits | Definition | 7 | When Set, this bit indicates that this is the last MultiRecord in the list. | 6:0 | This field indicates the MultiRecord format version. This field must be set to a value of 2h. |
| Bits | Definition | | | | | | | |
| 7 | When Set, this bit indicates that this is the last MultiRecord in the list. | | | | | | | |
| 6:0 | This field indicates the MultiRecord format version. This field must be set to a value of 2h. | | | | | | | |
| 2h | Implementation Specific | Record Length (RLEN): This field indicates the length of the PCI-SIG MultiRecord Body in bytes (i.e., the bytes of the MultiRecord excluding the 5-byte MultiRecord header that is common to all MultiRecords). | | | | | | |
| 3h | Implementation Specific | Record Checksum: This field is used to give the PCI-SIG MultiRecord Body a zero checksum (i.e., the modulo 256 sum of bytes from byte offset 5h to the end of this MultiRecord plus this checksum byte equals 0h). | | | | | | |
| 4h | Implementation Specific | Header Checksum: This field is used to give the PCI-SIG MultiRecord Header a zero checksum (i.e., the modulo 256 sum of offset 0h of the PCI-SIG MultiRecord Header through this checksum byte equals 0h). | | | | | | |
| PCI-SIG MultiRecord Body | | | | | | | | |
| 7h:5h | E8FFh | [PCI-SIG IANA Private Enterprise Number]: This field indicates the PCI-SIG's IANA Private Enterprise Number. This field must be set to a value of E8FFh (i.e., 59,647). | | | | | | |
| 8h | 0h | Version Number: This field indicates the version number of this MultiRecord. This field must be 0h in this version of the specification. | | | | | | |
| 9h | Implementation Specific | Number of Descriptors: This field indicates the number of descriptors in this MultiRecord. The value of 0h is reserved. | | | | | | |

| Bytes | Factory Default | Description |
|-------------------------|-------------------------|--|
| Implementation Specific | Implementation Specific | Descriptor 0: This field contains the first descriptor in this MultiRecord. |
| Implementation Specific | Implementation Specific | Descriptor 1: This field contains the second descriptor in this MultiRecord if Number of Descriptors is greater than one; otherwise, this field is not present. |
| ... | ... | ... |

Table 12-17 PCI-SIG MultiRecord Descriptor §

| Bytes | Factory Default | Description | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------|---------------------------------------|--|--------------|----------|-------------|----------|-----------|--------------------------------|-----------|-------------------------------|-----------|-------------------------------|-----------|-----------------------------|-----------|--------------|-----------|---------------------|-----------|------------------------------|-----------|---------------------------------------|-----------|---------------------------------------|---------------|----------|
| 1h:0h | Implementation Specific | <p>Descriptor Type: This field indicates the type of this descriptor.</p> <p>Bits:</p> <table> <tr> <td>15:13</td> <td>Reserved</td> </tr> <tr> <td>11:8</td> <td>Group ID</td> </tr> <tr> <td>0h</td> <td>PCI Express Base Specification</td> </tr> <tr> <td>1h</td> <td>PCI Express CEM Specification</td> </tr> <tr> <td>2h</td> <td>PCI Express M.2 Specification</td> </tr> <tr> <td>3h</td> <td>PCI Express SFF-8639 Module</td> </tr> <tr> <td>4h</td> <td>PCI Firmware</td> </tr> <tr> <td>5h</td> <td>PCI Express OCuLink</td> </tr> <tr> <td>6h</td> <td>PCI Express External Cabling</td> </tr> <tr> <td>7h</td> <td>PCI Express CopperLink Internal Cable</td> </tr> <tr> <td>8h</td> <td>PCI Express CopperLink External Cable</td> </tr> <tr> <td>Others</td> <td>Reserved</td> </tr> </table> <p>7:0 Sub-Type (Defined by Group ID owner)</p> | 15:13 | Reserved | 11:8 | Group ID | 0h | PCI Express Base Specification | 1h | PCI Express CEM Specification | 2h | PCI Express M.2 Specification | 3h | PCI Express SFF-8639 Module | 4h | PCI Firmware | 5h | PCI Express OCuLink | 6h | PCI Express External Cabling | 7h | PCI Express CopperLink Internal Cable | 8h | PCI Express CopperLink External Cable | Others | Reserved |
| 15:13 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11:8 | Group ID | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0h | PCI Express Base Specification | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1h | PCI Express CEM Specification | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2h | PCI Express M.2 Specification | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3h | PCI Express SFF-8639 Module | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4h | PCI Firmware | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5h | PCI Express OCuLink | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6h | PCI Express External Cabling | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7h | PCI Express CopperLink Internal Cable | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8h | PCI Express CopperLink External Cable | | | | | | | | | | | | | | | | | | | | | | | | | |
| Others | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2h | Implementation Specific | Descriptor Version Number: This field indicates the version number of this descriptor. This field must be 0h. | | | | | | | | | | | | | | | | | | | | | | | | |
| 3h | Implementation Specific | Descriptor Length: This field indicates the number of bytes in this descriptor. | | | | | | | | | | | | | | | | | | | | | | | | |
| Length N-1: 4h | Implementation Specific | Descriptor Data: This field contains the type-specific data associated with this descriptor where N is the total length in bytes of this descriptor. | | | | | | | | | | | | | | | | | | | | | | | | |

Each Group ID owner can define a list of sub-types with associated Descriptor Data. The Sub-Type field in the PCI-SIG MultiRecord Descriptor is 8 bits allowing for 256 sub-types.

Descriptor Data can be anything defined by the Group ID owner but should follow IPMI FRU conventions and should be kept as small as possible.

Types of FRU Information that are common across many form factors are defined in PCI-SIG MultiRecord Descriptors in this specification as part of Group ID 0h. Types of FRU Information that are not common across many form factors are defined in PCI-SIG MultiRecord Descriptors in other specifications as part of Group IDs other than 0h.

12.6.3 Common PCI-SIG MultiRecord Descriptors §

This section defines the PCI-SIG MultiRecord Descriptor sub-types for the PCI Express Base Specification group (i.e., Group ID 0h). These descriptors are common across many PCI Express-based adapters and components.

Table 12-18 Descriptor Sub-Types for Group ID 0h §

| Descriptor Sub-Type | Description |
|---------------------|-----------------------|
| 00h | Connector Subdivision |
| All other encodings | Reserved |

12.6.3.1 Connector Subdivision (Group ID 0h, Sub-Type 0h) §

In some use cases, the host requires the ability to discover connector subdivision combinations of an adapter's or component's upstream-facing connector so that the host can configure the subdivision of the Downstream Port connected to the adapter or component. A couple of examples include:

- a. Two x8 Ports on an adapter with a x16 upstream-facing connector.
- b. Four M.2 x4 connectors on a CEM card carrier with a x16 upstream-facing connector.

This section defines the FRU descriptors that describe the adapter's or component's supported connector subdivision combination(s). If only one connector subdivision combination is supported, then no Connector Subdivision Combinations Descriptor (refer to § Table 12-19) is required. If more than one connector subdivision combination is supported and the implementation supports the PCI-SIG MultiRecord, then a Connector Subdivision Combinations Descriptor with an instance of the **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** (refer to § Table 12-20) for each supported connector subdivision combination must be present.

The **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** at the lowest offset in the Connector Subdivision Combinations Descriptor should include the most common expected usage and must be the default **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑**. This is unless a form factor specific method is applied (e.g., the DUALPORTEN# sideband signal defined by the [SFF-8639]) or the host commands the form factor to select an alternate **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** using a mechanism outside the scope of this specification. The default **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** is defined as the adapter configuration if no control function is sent to select an alternate mode of operation. The adapter or component must revert to the default **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** only on a Cold Reset.

Each Lane on the upstream-facing connector within each connector subdivision combination that is connected to a Port or another connector on the adapter or component must be described within a single **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑**. Lanes on the upstream-facing connector that are not connected to a Port or another connector on the adapter or component must not be included in the Connector Subdivision Combinations Descriptor. The Lanes associated with a **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** may be organized as a single subdivision or multiple subdivisions. Each subdivision within each **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑** must be listed in ascending order starting from the adapter's or component's lowest-numbered Lane.

The lowest-numbered Lane must always be with respect to the adapter or component's designated lane 0 without lane reversal. The starting lane number of any subdivision within a **↓↑Connector Subdivision Descriptor↑↓↑Connector Subdivision Descriptor↑**

Subdivision Descriptor must be less than or equal (if the subdivision width is x1) to the ending lane number of the respective subdivision.

The details of how connector subdivision information is passed from an out-of-band agent (e.g., a Baseboard Management Controller) responsible for reading this information to the platform entity (e.g., the platform BIOS) responsible for configuring the appropriate Downstream Port(s) is implementation specific.

It is recommended that adapters include the Connector Subdivision Combinations Descriptor, even if only a single width configuration is supported, because it enables system firmware to determine degraded link width conditions in situations where an adapter advertises a value in the Maximum Link Width field in the Link Capabilities Register that is less than or greater than the physical adapter connector edge connectivity (e.g., a x8 capable component on an adapter that only connects x4 to the adapter card edge).

Table 12-19 Connector Subdivision Combinations Descriptor §

| Byte Offset(s) | Factory Default | Description |
|-------------------------|-------------------------|---|
| 1h:0h | 0001h | <p>Descriptor Type: This field indicates the type of this descriptor.</p> <p>Bits:</p> <ul style="list-style-type: none"> 15:12 Reserved 11:8 Group ID = 0h (PCI Express Base Specification) 7:0 Sub-Type = 00h (Connector Subdivision) |
| 2h | 01h | Descriptor Version Number: This field indicates the version number of this descriptor. This field must be 0h. |
| 3h | 04h | Header Length: This field indicates the length of the header portion of this descriptor (meaning from offset 0h until but not including the Connector Subdivision 0 field). This serves the purpose of being able to insert future bytes after the Descriptor Length and before the Connector Subdivision list if new useful attributes arise. |
| 4h | Implementation Specific | Descriptor Length: This field indicates the length of this descriptor in bytes. |
| Implementation Specific | Implementation Specific | Connector Subdivision 0: This field contains the first connector subdivision. |
| Implementation Specific | Implementation Specific | Connector Subdivision 1: This field contains the second connector subdivision, if applicable. If there is only one connector subdivision, then this field is not present. |
| ... | ... | ... |

The ~~↓↑Connector Subdivision Descriptor~~ ~~↑↑Connector Subdivision Descriptor~~ is shown in § Table 12-20 . An upstream PCIe connector is made up of one or more subdivisions. Each subdivision is routed to a Port or another connector on the adapter or component. The ~~↓↓Connector Subdivision Descriptor~~ ~~↓↑Connector Subdivision Descriptor~~ indicates the number of subdivisions and the starting and ending Lane numbers for each subdivision that make up each supported connector subdivision combination.

Table 12-20 ~~↓↓Connector Subdivision Descriptor~~ ~~↑↑Connector Subdivision Descriptor~~ §

| Byte Offset(s) | Factory Default | Description |
|----------------|-------------------------|---|
| 0h | Implementation Specific | Descriptor Length: This field indicates the length of this descriptor in bytes. |
| 1h | Implementation Specific | Number of Subdivisions: This field indicates the number of subdivisions in this descriptor. The permitted range of values is from 1 (e.g., 1 subdivision of maximum supported width) to the number |

| Byte Offset(s) | Factory Default | Description |
|----------------|-------------------------|--|
| | | of individual lanes used at the maximum supported width. All other values are Reserved. As an example, a x16 capable adapter can have values from 1 (e.g., one subdivision of x16 width) to 16 (e.g., 16 subdivisions of x1 width). |
| 2h | Implementation Specific | <p>Subdivision 0: This field indicates the starting and ending lanes of the first subdivision of this descriptor.</p> <p>Bits:</p> <ul style="list-style-type: none"> 7:4 Starting Lane is the lowest-numbered Lane of the first subdivision of this descriptor. 3:0 Ending Lane is the highest numbered Lane of the first subdivision of this descriptor. |
| 3h | Implementation Specific | <p>Subdivision 1: This field indicates the starting and ending lanes of the second subdivision of this descriptor.</p> <p>Bits:</p> <ul style="list-style-type: none"> 7:4 Starting Lane is the lowest-numbered Lane of the second subdivision of this descriptor. 3:0 Ending Lane is the highest numbered Lane of the second subdivision of this descriptor. |
| ... | ... | ... |

Example Connector Subdivision Combinations Descriptor for an Adapter Supporting 3 ↓↑Connector Subdivision Descriptors↓↑Connector Subdivision Descriptor s↑ :

1 Port of x16 Lanes, 2 Ports of x8 Lanes each, 4 Ports of x4 Lanes each

↓↑Connector Subdivision Descriptor:↓↑Connector Subdivision Descriptor :↑ 0 (the default)

Number of Subdivisions: 1

Starting Lane of the first subdivision: 0; Ending Lane of the first subdivision: 15

↓↑Connector Subdivision Descriptor:↓↑Connector Subdivision Descriptor :↑ 1

Number of Subdivisions: 2

Starting Lane of the first subdivision: 0; Ending Lane of the first subdivision: 7

Starting Lane of second subdivision: 8; Ending Lane of second subdivision: 15

↓↑Connector Subdivision Descriptor:↓↑Connector Subdivision Descriptor :↑ 2

Number of Subdivisions: 4

Starting Lane of the first subdivision: 0; Ending Lane of the first subdivision: 3

Starting Lane of second subdivision: 4; Ending Lane of second subdivision: 7

Starting Lane of third subdivision: 8; Ending Lane of third subdivision: 11

Starting Lane of forth subdivision: 12; Ending Lane of forth subdivision: 15

12.7 Out-of-Band Management Control Mechanism §

ECN: Base
6.3
PCIe-MI $\Delta \triangleleft \triangleright$

After inventory, This section defines the optional PCI Express Management Interface (PCIe-MI). PCIe-MI is an architecture and discovery command set for message-based management of a PCIe-MI Component (e.g., an adapter or component). The following management functionality is defined.

- Host CPU-agnostic and operating system-agnostic out-of-band management.
- A common management interface used across all in-band management transports and out-of-band management transports.
- Commands to perform management operations such as the following.
 - Discover the PCIe-MI Components that are connected to the host system.
 - Discover the properties and capabilities of each PCIe-MI Component.
 - Monitor runtime changes to the PCIe-MI Component such as changes to the speed and width of any Link on the PCIe-MI Component.
 - PCIe-specific configuration of PCIe-MI Components such as the following.
 - Get and set Configuration Space bits and fields using out-of-band management transports.
 - Get and set a Function's Segment Number, Bus Number, and/or Device Number.
 - Get and set connector subdivisions.

The management messages are referred to as PCIe-MI Messages. The two types of PCIe-MI Messages are PCIe-MI Commands and PCIe-MI Completion.

- A PCIe-MI Command is transmitted from a PCIe-MI Commander, such as a Baseboard Management Controller (BMC) or in-band management system software, to a PCIe-MI Completer (e.g., an interposer, adapter, MCTP endpoint or component, control function(s) a Function that supports MMPT) that specifies a management operation to be performed.
- In order to provide the results of a PCIe-MI Command, a PCIe-MI Completion must be transmitted from the PCIe-MI Completer that received the PCIe-MI Command to the PCIe-MI Commander that transmitted the PCIe-MI Command.

PCIe-MI is supported using out-of-band management transports (i.e., independent of system software running on a host CPU) from a management controller (see the [MCTP Base Specification - <https://www.dmtf.org/dsp/DSP0236>]) such as a BMC and is supported using in-band management transports (i.e., from system software running on a host CPU). The supported out-of-band management transports for PCIe-MI include any MCTP-based intercommunications between a management controller (e.g., a BMC) and a managed device (i.e., a PCIe-MI Component) as described in the [Platform Management Communications Infrastructure (PMCI) Architecture Whitepaper - <https://www.dmtf.org/dsp/DSP2015>]. The supported in-band management transports for PCIe-MI include the following.

- PCIe-MI over MMPT (see § Section 6.35.1.4 and the PCI Code and ID Assignment Specification).
- Any MCTP-based intercommunications between host platform software (i.e., system software running on a host CPU) and a management controller (i.e., a PCIe-MI Component) as described in the [Platform Management Communications Infrastructure (PMCI) Architecture Whitepaper - <https://www.dmtf.org/dsp/DSP2015>].

Base 6.4 vs Base 6.3

↑↑PCIe Components that implement PCIe-MI over MCTP must comply to the [MCTP Base Specification - <https://www.dmtf.org/dsp/DSP0236>], the [PCIe-MI over MCTP Binding Specification - <https://www.dmtf.org/dsp/DSP0291>], and any applicable MCTP transport or interface specifications such as the following:↑

- ↑↑The [MCTP SMBus/I2C Binding Specification - <https://www.dmtf.org/dsp/DSP0237>].↑
- ↑↑The [MCTP PCIe VDM Binding Specification - <https://www.dmtf.org/dsp/DSP0238>].↑
- ↑↑The [MCTP I3C Binding Specification - <https://www.dmtf.org/dsp/DSP0233>].↑
- ↑↑The [MCTP USB Binding Specification - <https://www.dmtf.org/dsp/DSP0283>].↑
- ↑↑The [MCTP MMBI Binding Specification - <https://www.dmtf.org/dsp/DSP0284>].↑
- ↑↑The [MCTP Host Interface Specification - <https://www.dmtf.org/dsp/DSP0256>].↑

↑↑For PCIe-MI over MCTP, PCIe-MI Messages are considered PCIe-MI Request messages per the [PCIe-MI over MCTP Binding Specification - <https://www.dmtf.org/dsp/DSP0291>]. Therefore, the Tag Owner bit defined by the [PCIe-MI over MCTP Binding Specification - <https://www.dmtf.org/dsp/DSP0291>] is Set in PCIe-MI Messages (i.e., PCIe-MI Commands and PCIe-MI Completions), and the value of the Message Tag field defined by the [PCIe-MI over MCTP Binding Specification - <https://www.dmtf.org/dsp/DSP0291>] always originates from the PCIe-MI Commander or PCIe-MI Completer that is transmitting the PCIe-MI Command or PCIe-MI Completion . Additionally, since there are no PCIe-MI Response messages, there are no timing requirements related to PCIe-MI Response messages. For example, there is no request-to-response timing requirement and instead, the PCIe-MI Command to PCIe-MI Completion time must adhere to the time indicated by the Maximum Command-To-Completion Time field in § Table 12-39 .↑

↑↑§ Figure 12-18 shows the PCIe-MI protocol layers on a PCIe-MI Component for the transports described in this section. PCIe-MI ↑ may ↑ also ↑ be ↑ required. Control commands ↑ transported over transport layers that ↑ are ↑ recommended ↑ not shown or transported over MCTP over physical layers with MCTP bindings that are not shown. The details related to ↑ utilize ↑ any transport or physical layer with an MCTP binding not defined by this specification are implementation specific. Note that ↑ the ↑ Management Control Transport ↑ MCTP Host Interface as defined by the [MCTP Host Interface Specification - <https://www.dmtf.org/dsp/DSP0256>] may also be utilized for use cases such as MCTP over MMBI (not shown in § Figure 12-18).↑

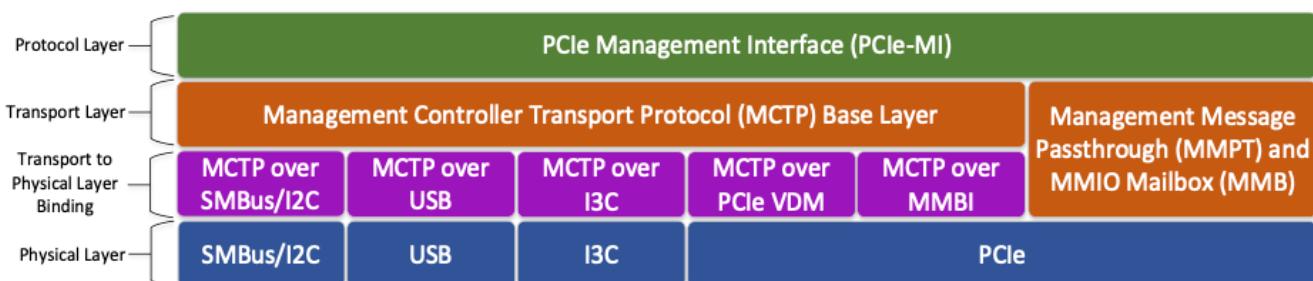


Figure 12-18 ↑↑PCIe-MI Component↑ Protocol ↑↑(MCTP).↓ ↑↑Layers↑ §

12.7.1 ↑↑PCIe-MI Architectural Model↑ §

ECN: Base 6.3
PCIe-MI△◀▷

↑↑A PCIe-MI Component is a collection of one or more Functions, such as a component (e.g., a Root Complex, Switch, or embedded Endpoint) or adapter (e.g., a network adapter or a GPU on a CEM add-in card, or an NVMe drive in a U.2 or EDSFF form factor) and has the following characteristics.↑

- The ↑↑rationale↓ ↑↑PCIe-MI Component must contain one or more non IOV Functions and/or Physical Functions.↑

Base 6.4 vs Base 6.3

- The PCIe-MI Component must support PCIe-MI on one or more PCIe-MI Completers . Examples of PCIe-MI Completers include the following.
 - A Function supporting PCIe-MI over MMPT.
 - An MCTP endpoint supporting PCIe-MI over MCTP over one or more of the following.
 - PCIe VDM.
 - SMBus/I2C.
 - I3C.
 - USB.
 - MMBI.
- A PCIe-MI Port must be one of the following.
 - A PCIe Upstream Port.
 - A PCIe Downstream Port.
 - A 2-Wire interface port.
 - A USB port.
- A PCIe-MI Component must contain one or more PCIe ports (i.e., Upstream Ports or Downstream Ports).
- A PCIe-MI Component contains zero or more non-PCIe PCIe-MI Ports.
- Each PCIe-MI Port on the PCIe-MI Component must have a PCIe-MI Port Identifier that is unique within the PCIe-MI Component (see § Section 12.7.3.1.5).
 - PCIe-MI Port Identifiers must be in the range 0 to N minus 1, where N is the number of PCIe-MI Ports on the PCIe-MI Component.
 - PCIe-MI Port Identifiers must not change value unless main power to the PCIe-MI Component is removed.
- Each PCIe-MI Port is exposed or hidden from the host system as indicated by the PCIe-MI Port Exposed bit in § Table 12-24 .
 - Examples of exposed/hidden ports include the following.
 - PCIe ports:
 - A PCIe-MI Component supports a well-supported higher-level protocol versus low-level register-based solutions, fixed components like I2C I/O expanders. connector subdivision with a x8 PCIe port on lanes 0 to 7 and a connector subdivision with two x4 PCIe ports where one x4 PCIe port is on lanes 0 to 3 and the other x4 PCIe port is on lanes 4 to 7.
 - The x8 connector subdivision is selected.
 - In this example, the PCIe-MI Component has three PCIe ports where the x8 PCIe port is exposed, and each x4 PCIe port is hidden.
 - USB port:
 - A PCIe-MI Component supports a USB port on the JTAG pins of a CEM add-in card connector.
 - If the JTAG pins are configured for USB, then the USB port is exposed.
 - If the JTAG pins are configured for JTAG, then the USB port is hidden.
- Each PCIe-MI Port that is a PCIe Upstream Port or discrete wires. PCIe Downstream Port contains zero or more Functions supporting PCIe-MI over MMPT.
 - Each Function supporting PCIe-MI over MMPT contains an MMPT PCIe-MI Completer .
- Each PCIe-MI Port contains zero or more MCTP physical addresses supporting PCIe-MI.

- ~~↑↑Each MCTP physical address supporting PCIe-MI must contain one or more PCIe-MI Completers (i.e., MCTP endpoints, see the [MCTP Base Specification - <https://www.dmtf.org/dsp/DSP0236>]):↑
▪ ~~↑↑Supporting more than one MCTP endpoint per MCTP physical address requires the PCIe-MI Component to support MCTP bridging (see the [MCTP Base Specification - <https://www.dmtf.org/dsp/DSP0236>]).↑~~~~
- ~~↑↑The PCIe-MI Component may support a FRU Information Device accessible in SMBus/I2C mode on the 2-Wire interface port (see § Section 12.6.1).↑~~

12.7.1.1 ~~↑↑Operational Times~~ §

*ECN: Base 6.3
PCIe-MI* △▷

~~↑↑A PCIe-MI Completer that is able to receive PCIe-MI Commands , process PCIe-MI Commands successfully, and transmit PCIe-MI Completions is operational.↑~~

~~↑↑For PCIe-MI over MMPT, a PCIe-MI Completer must be operational whenever MMPT is operational.↑~~

~~↑↑For PCIe-MI over MCTP, a PCIe-MI Completer is operational under the following situations.↑~~

- ~~↑↑If a reset that causes a PCIe-MI Completer Reset (see § Section 12.7.1.2) defines an assertion and a deassertion condition (e.g., SMBRST# as defined in the [SNIA SFF-TA-1009 Enterprise and Datacenter Standard Form Factor Pin and Signal Specification - <https://www.snia.org/sff>] or PERST#), then the PCIe-MI Completer must become operational within 1 s from the de-assertion of that reset unless the PCIe-MI Completer is undergoing a subsequent PCIe-MI Completer Reset.↑~~
- ~~↑↑If a reset that causes a PCIe-MI Completer Reset (see § Section 12.7.1.2) defines an assertion condition but no deassertion condition (e.g., application of power or a Target Reset Pattern as defined by the [MIPI I3C Basic Specification - <https://www.mipi.org>] with RSTACT Defining Byte value 2), then the PCIe-MI Completer must become operational within 1 s from the assertion of that reset unless the PCIe-MI Completer is undergoing a subsequent PCIe-MI Completer Reset.↑~~
- A ~~↑↑future revision~~ ~~↑↑PCIe-MI Completer must become operational 1 s after main power is applied unless the PCIe-MI Completer is undergoing a PCIe-MI Completer Reset due to a reason other than application of main power (see § Section 12.7.1.2).↑~~
- ~~↑↑A PCIe-MI Completer may become operational with only auxiliary power supplied.↑~~
 - ~~↑↑If a PCIe-MI Completer supports being operational with only auxiliary power supplied, then the PCIe-MI Completer must become operational 1 s after only auxiliary power is applied unless the PCIe-MI Completer is undergoing a PCIe-MI Completer Reset due to a reason other than application of only auxiliary power.↑~~
- ~~↑↑If main power is applied, then application or removal of auxiliary power must have no impact on the operational state of the PCIe-MI Completer .↑~~
- ~~↑↑If the PCIe-MI Completer supports operation with only auxiliary power applied and auxiliary power is applied, then application or removal of ~~↑↑~~main power must have no impact on the operational state of the PCIe-MI Completer .↑~~
- ~~↑↑Once a PCIe-MI Completer becomes operational, the PCIe-MI Completer must remain operational unless the PCIe-MI Completer undergoes a PCIe-MI Completer Reset.↑~~
- ~~↑↑PCIe-MI over MCTP must be operational during a Conventional Reset or a Function Level Reset unless the PCIe-MI Completer is one of the following.↑~~
 - ~~↑↑A PCIe VDM MCTP endpoint or MMBI MCTP endpoint on a Function that is undergoing a Conventional Reset or Function Level Reset.↑~~
 - ~~↑↑A PCIe-MI Completer that is undergoing a PCIe-MI Completer Reset due to a reason other than a Conventional Reset or a Function Level Reset.↑~~

12.7.1.2 ~~PCIe-MI Completer Reset~~ §

ECN: Base 6.3
PCIe-MI Δ

~~If a PCIe-MI Completer undergoes a PCIe-MI Completer Reset, then the PCIe-MI Completer must abort the processing of any PCIe-MI Commands, must not transmit a PCIe-MI Completion for PCIe-MI Commands whose processing has been aborted, and should abort the transmission of any PCIe-MI Messages in progress. If the transmission of a PCIe-MI Message for PCIe-MI over MCTP is aborted due to a PCIe-MI Completer Reset, then the transmission should be aborted on an MCTP packet boundary. The following rules apply to PCIe-MI Completer Resets. The impact of any reset not listed in this specification will define this section is implementation specific.~~

- ~~When power is applied to the MCTP-based control mechanism, then a PCIe-MI Completer, then a PCIe-MI Completer Reset must occur.~~
- ~~If the PCIe-MI Completer is a PCIe VDM MCTP endpoint, MMPT-capable Function, or MMBI-capable Function, then a Conventional Reset or a Function Level Reset must cause a PCIe-MI Completer Reset.~~
- ~~If the PCIe-MI Completer is an MCTP endpoint, then a Conventional Reset or a Function Level Reset must not cause a PCIe-MI Completer Reset unless the PCIe-MI Completer is a PCIe VDM MCTP endpoint or MMBI MCTP endpoint on a Function that is undergoing the Conventional Reset or Function Level Reset.~~
- ~~If the PCIe-MI Completer is an MCTP endpoint on a 2-Wire interface port and the 2-Wire clock is held low for longer than T_{dcl} (see § Table 12-14), then a PCIe-MI Completer Reset must occur.~~
- ~~If the PCIe-MI Completer is an MCTP endpoint on a 2-Wire interface port and a form factor-specific 2-Wire reset occurs (e.g., assertion of SMBRST# as defined in the [SNIA SFF-TA-1009 Enterprise and Datacenter Standard Form Factor Pin and Signal Specification - <https://www.snia.org/sff>]), then a PCIe-MI Completer Reset must occur.~~
- ~~If the PCIe-MI Completer is an MCTP endpoint on a 2-Wire interface port, then a PCIe-MI Completer Reset must cause the PCIe-MI Completer to do the following in the order listed.~~
 1. ~~Stop transmitting any PCIe-MI Message in flight within 5 ms from the assertion of the PCIe-MI Completer Reset as follows.~~
 - ~~If in SMBus mode, generate a STOP condition to signal the end of the transmission (see the [SMBus Specification - <https://www.smbus.org>]).~~
 - ~~If in I3C mode, set the T bit to 1b to signal the end of the transmission (see the [MIPI I3C Basic Specification - <https://www.mipi.org>]).~~
 2. ~~Transition the transmitter to the bus idle condition (refer to the [SMBus Specification - <https://www.smbus.org>] and the [MIPI I3C Basic Specification - <https://www.mipi.org>]).~~
 - ~~Keep the transmitter in the bus idle condition and ignore any incoming traffic until the PCIe-MI Completer is operational again (see § Section 12.7.1.1).~~
 3. ~~Reset the 2-Wire interface port physical layer.~~
 - ~~If the 2-Wire interface port is in I3C Mode, then the 2-Wire interface port must switch to SMBus mode as part of resetting the 2-Wire interface port physical layer due to the PCIe-MI Completer Reset, unless otherwise specified (e.g., the PCIe-MI Completer Reset is due to an I3C Target Reset Pattern with RSTACT Defining Byte value 2).~~
- ~~If the PCIe-MI Completer is an MCTP endpoint on a 2-Wire interface port in I3C mode, then the Target Reset Pattern as defined by the [MIPI I3C Basic Specification - <https://www.mipi.org>] behaves as follows.~~
 - ~~All Target Reset Patterns must not change the 2-Wire interface port's mode to SMBus mode (i.e., the 2-Wire interface port must remain in I3C mode).~~
 - ~~The default cases for the Target Reset Pattern must reset the I3C physical layer, must not cause a PCIe-MI Completer Reset, and must not reset the I3C Dynamically Assigned Address.~~

- ↑↑For non-default cases for the Target Reset Pattern, the RSTACT CCC as defined by the [MIPI I3C Basic Specification - <https://www.mipi.org>] may be supported and if supported, behaves as follows.↑
 - ↑↑Defining Byte value 0 must not cause a PCIe-MI Completer Reset and must have no effect.↑
 - ↑↑Defining Byte value 1 must behave the same as the default cases for the Target Reset Pattern.↑
 - ↑↑Defining Byte value 2 must cause a PCIe-MI Completer Reset and must reset the I3C Dynamically Assigned Address.↑
- ↑↑If the PCIe-MI Completer is an MCTP endpoint on a USB 2.0 port and the port is reset using USB Reset signaling as defined in the [Universal Serial Bus Specification, Revision 2.0 – <https://www.usb.org>], then a PCIe-MI Completer Reset must occur.↑
- ↑↑A PCIe-MI Completer Reset of a given PCIe-MI Completer must not affect the operation of any other entity, including other PCIe-MI Completers , outside of the PCIe-MI Completer undergoing the PCIe-MI Completer Reset.↑

12.7.1.3 Example PCIe-MI Component §

*ECN: Base 6.3
PCIe-MI* Δ ◀ ▷

IMPLEMENTATION NOTE: EXAMPLE PCIE-MI COMPONENT §

↑↑↑§ Figure 12-19 shows an example PCIe-MI Component with the following properties.↑

- ↑↑↑Two PCIe ports with PCIe-MI Port Identifiers of 0 and 1.↑
 - ↑↑↑PCIe-MI Port Identifier 0.↑
 - ↑↑↑One PCIe Function with a Function Number of 0.↑
 - ↑↑↑Function 0 supports PCIe-MI over MMPT.↑
 - ↑↑↑Function 0 supports PCIe-MI over MCTP over PCIe VDM.↑
 - ↑↑↑PCIe-MI Port Identifier 1.↑
 - ↑↑↑Two PCIe Functions with Function Numbers of 0 and 1.↑
 - ↑↑↑Function 0 and 1 support PCIe-MI over MMPT.↑
 - ↑↑↑Function 0 supports PCIe-MI over MCTP over PCIe VDM.↑
 - ↑↑↑Each Function that supports PCIe-MI over MMPT may support PCIe-MI over MCTP over MMBI instead of or in addition to PCIe-MI over MMPT or may support neither PCIe-MI over MMPT nor PCIe-MI over MCTP over MMBI.↑
 - ↑↑↑Each PCIe port is an Upstream Port (e.g., a Switch Upstream Port or an Upstream Port on an Endpoint) or a Downstream Port (e.g., a Root Port on a Root Complex or a Switch Downstream Port).↑
- ↑↑↑One 2-Wire interface port with a PCIe-MI Port Identifier of 2 with SMBus mode addresses shown in 8-bit/7-bit format.↑
 - ↑↑↑One FRU Information Device at physical address A6h/53h in SMBus/I2C mode only (see § Section 12.6.1).↑
 - ↑↑↑A primary MCTP physical address at address 3Ah/1Dh in SMBus mode.↑
 - ↑↑↑Two MCTP endpoints (e.g., to support redundant BMCs).↑
 - ↑↑↑Each MCTP endpoint is operational only while main power is applied.↑
 - ↑↑↑A secondary MCTP physical address at address D4h/6Ah in SMBus mode.↑
 - ↑↑↑One MCTP endpoint.↑
 - ↑↑↑The MCTP endpoint is operational while auxiliary power only, main power only, or both is applied.↑
 - ↑↑↑The ability to become operational with only auxiliary power applied allows a BMC to perform tasks before applying main power such as configuring connector subdivisions so that the Links train to their configured widths once main power is applied.↑
 - ↑↑↑Optionally supports the ability to transition to I3C mode.↑

Base 6.4 vs Base 6.3

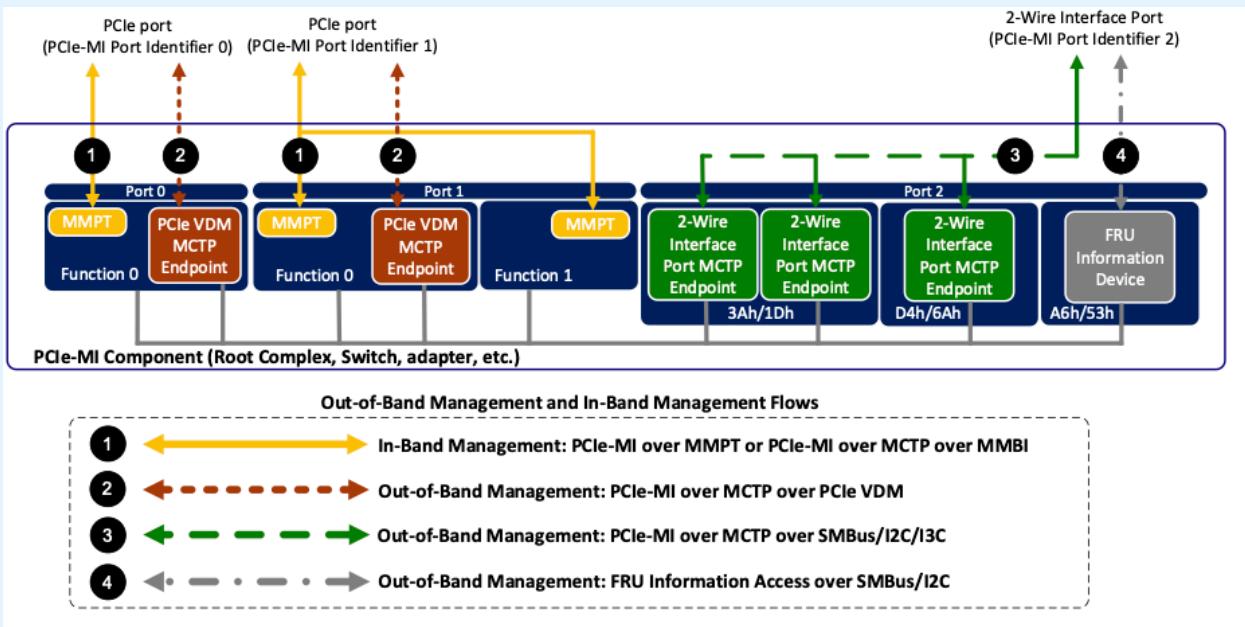


Figure 12-19 *Example PCIe-MI Components* §

12.7.1.4  Example PCIe-MI Management Traffic Flows

*ECN: Base 6.3
PCIe-MI* 

IMPLEMENTATION NOTE: EXAMPLE PCIE-MI MANAGEMENT TRAFFIC FLOWS §

~~Additional out of band~~ ~~Figure 12-20 shows example PCIe-MI Message flows.~~

- ~~In-band management system software (e.g., the operating system or UEFI firmware running on a host CPU) communicates with the PCIe-MI Component via PCIe-MI over MMPT or PCIe-MI over MCTP over MMBI as shown in management flow number 1 (see § Section 6.35.1.4 and the PCI Code and ID Assignment Specification).~~
- ~~A BMC communicates with the PCIe-MI Component via PCIe-MI over MCTP (see the [PCIe-MI over MCTP Binding Specification - <https://www.dmtf.org/dsp/DSP0291>] over PCIe VDM (see the [MCTP PCIe VDM Binding Specification - <https://www.dmtf.org/dsp/DSP0238>]) as shown in management flow number 2.~~
 - ~~PCIe-MI Commands are transmitted peer-to-peer using Route by ID from the BMC to the PCIe-MI Component.~~
 - ~~PCIe-MI Completions are transmitted peer-to-peer using Route by ID from the PCIe-MI Component to the BMC.~~
 - ~~The Root Complex also transmits MCTP control methods messages to the PCIe-MI Component using Broadcast from Root Complex as defined by the [MCTP PCIe VDM Binding Specification - <https://www.dmtf.org/dsp/DSP0238>] and the PCIe-MI Component responds using Route to Root Complex (not shown in the figure).~~
 - ~~The Root Complex may contain an MCTP bridge used to route MCTP messages based on MCTP endpoint ID as defined by the [MCTP Base Specification - <https://www.dmtf.org/dsp/DSP0236>] and the [MCTP PCIe VDM Binding Specification - <https://www.dmtf.org/dsp/DSP0238>] (not shown in the figure).~~
 - ~~A BMC communicates with each of the PCIe-MI Components via PCIe-MI over MCTP over a sideband bus that supports MCTP such as SMBus/I2C, I3C, USB, etc. (see the applicable MCTP physical layer binding specification) as shown in management flow number 3.~~

Base 6.4 vs Base 6.3

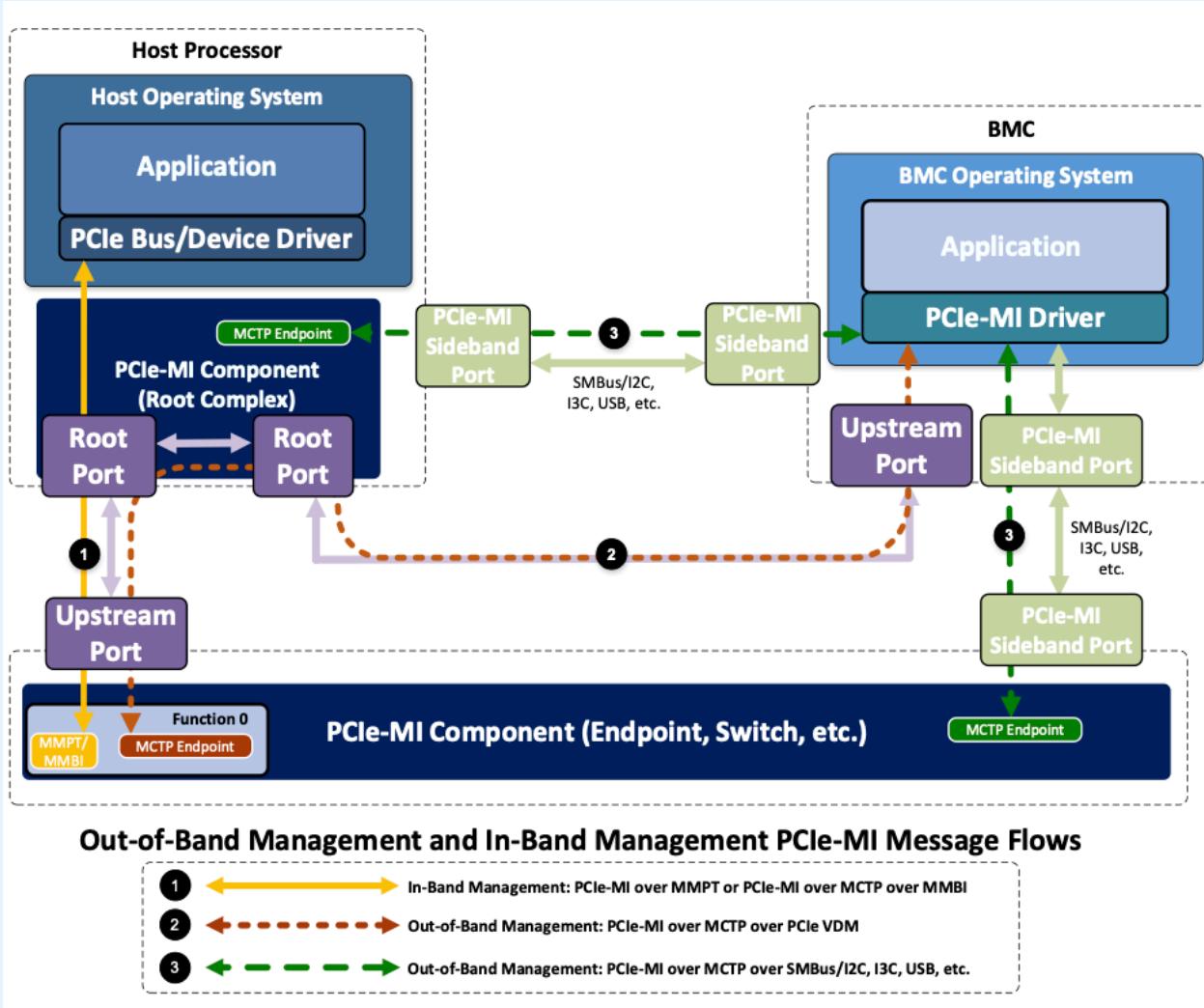


Figure 12-20 ↑↑Example PCIe-MI Message Flows↑↑

ECN: Base 6.3
PCIe-MI△↔

12.7.2 ↑↑PCIe-MI Message Format↑↑ §

↑↑The format of a PCIe-MI Message is shown in § Figure 12-21 . The PCIe-MI Message is a PCIe-MI Command or a PCIe-MI Completion as defined by the PCIe-MI Message Type (PMT) field. The endianness of all PCIe-MI Message fields is little endian. The PCIe-MI Message format is common between PCIe-MI over MCTP and PCIe-MI over MMPT.↑↑

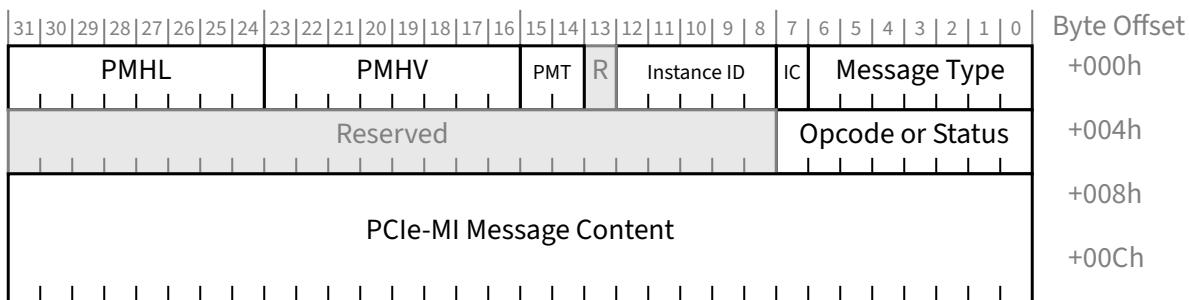


Figure 12-21 ¶PCIe-MI Message Format§

Base 6.4 vs Base 6.3

↑↑The format of a PCIe-MI Command is shown in § Table 12-21 .↑

Table 12-21 ¶PCIe-MI Command§

| ↑↑Byte Location↑ | ↑↑Field Description↑ | | | | | | | | | |
|----------------------------------|--|----------|----------------|--------|-----------------------|--------|--------------------|-----------|-------------|--|
| ↑↑PCIe-MI Message Header↑ | | | | | | | | | | |
| | ↑↑Message Type Information↑ : ↑↑This field specifies information about the message type.↑ | | | | | | | | | |
| ↑↑Bits↑ | ↑↑Description↑ | | | | | | | | | |
| ↑↑1:0↑ | <p>↑↑PCIe-MI Message Type (PMT)↑ : ↑↑This field should be set to 01b to specify the PCIe-MI Message is a PCIe-MI Command . If a PCIe-MI Completer receives a PCIe-MI Message with this field set to 01b, then the PCIe-MI Completer must handle the PCIe-MI Message as a PCIe-MI Command ; otherwise, an Invalid Input Error Completion with the Error Location field indicating this field must be returned (see § Section 12.7.2.2.1).↑</p> <table border="1"> <thead> <tr> <th>↑↑Value↑</th> <th>↑↑Description↑</th> </tr> </thead> <tbody> <tr> <td>↑↑00b↑</td> <td>↑↑PCIe-MI Completion↑</td> </tr> <tr> <td>↑↑01b↑</td> <td>↑↑PCIe-MI Command↑</td> </tr> <tr> <td>↑↑Others↑</td> <td>↑↑Reserved↑</td> </tr> </tbody> </table> | ↑↑Value↑ | ↑↑Description↑ | ↑↑00b↑ | ↑↑PCIe-MI Completion↑ | ↑↑01b↑ | ↑↑PCIe-MI Command↑ | ↑↑Others↑ | ↑↑Reserved↑ | |
| ↑↑Value↑ | ↑↑Description↑ | | | | | | | | | |
| ↑↑00b↑ | ↑↑PCIe-MI Completion↑ | | | | | | | | | |
| ↑↑01b↑ | ↑↑PCIe-MI Command↑ | | | | | | | | | |
| ↑↑Others↑ | ↑↑Reserved↑ | | | | | | | | | |
| ↑↑13↑ | ↑↑Reserved↑ | | | | | | | | | |
| ↑↑12:8↑ | ↑↑Instance Identifier (Instance ID)↑ : ↑↑This field specifies the instance of the PCIe-MI Command . See § Section 12.7.3 for additional details.↑ | | | | | | | | | |
| ↑↑7↑ | ↑↑Integrity Check (IC)↑ : ↑↑For PCIe-MI over MCTP, this bit is Clear per the [PCIe-MI over MCTP Binding Specification - https://www.dmtf.org/dsp/DSP0291]. For PCIe-MI over MMPT, this bit is not applicable and should be Clear. If a PCIe-MI Completer receives a PCIe-MI Message with this bit Set, then an Invalid Input Error Completion with the Error Location field indicating this bit must be returned (see § Section 12.7.2.2.1).↑ | | | | | | | | | |
| ↑↑6:0↑ | ↑↑Message Type↑ : ↑↑For PCIe-MI over MCTP, this field set to 9 specifies a PCIe-MI Message per the [PCIe-MI over MCTP Binding Specification - https://www.dmtf.org/dsp/DSP0291].↑ | | | | | | | | | |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|---|---|---|
| | ↑↑Bits↑ | ↑↑Description↑ |
| | | ↑↑For PCIe-MI over MMPT, this field set to 9 specifies a PCIe-MI Message (see the PCI Code and ID Assignment Specification). If a PCIe-MI Completer receives a PCIe-MI Message over MMPT with this field set to a value other than 9, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned (see § Section 12.7.2.2.1.).↑ |
| ↑↑2↑ | ↑↑PCIe-MI Message Header Version (PMHV)↑ | ↑↑This field specifies the version of the PCIe-MI Message Header. This field should be cleared to 0 for this version of the specification. See § Section 12.7.3 for additional details.↑ |
| ↑↑3↑ | ↑↑PCIe-MI Message Header Length (PMHL)↑ | ↑↑This field specifies the length of the PCIe-MI Message Header in bytes. This field should be set to 8 for this version of the specification. See § Section 12.7.3 for additional details.↑ |
| ↑↑4↑ | ↑↑Opcode↑ | ↑↑This field specifies the Opcode of the PCIe-MI Command (see § Table 12-27). If a PCIe-MI Completer receives a PCIe-MI Command with an unsupported Opcode, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned (see § Section 12.7.2.2.1.).↑ |
| ↑↑7:5↑ | ↑↑Reserved↑ | |
| ↑↑PCIe-MI Command Content↑ | | |
| ↑↑N:7:8↑ | ↑↑PCIe-MI Command Content↑ | ↑↑This optional field specifies PCIe-MI Command -specific content as defined in the following sections, where N is the size of this field in bytes. If N is 0, then this field is not present.↑ |
| ↑↑The format of a PCIe-MI Completion is shown in § Table 12-24 .↑ | | |
| <i>Table 12-24 ↑↑PCIe-MI Completion↑</i> § | | |
| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
| ↑↑PCIe-MI Message Header↑ | | |
| | ↑↑Message Type Information↑ : ↑↑This field indicates information about the message type.↑ | |
| | ↑↑Bits↑ | ↑↑Description↑ |
| ↑↑1:0↑ | ↑↑PCIe-MI Message Type (PMT)↑ | ↑↑This field must be cleared to 00b to indicate the PCIe-MI Message is a PCIe-MI Completion .↑ |
| | ↑↑Value↑ | ↑↑Description↑ |
| ↑↑15:14↑ | ↑↑00b↑ | ↑↑PCIe-MI Completion↑ |
| | ↑↑01b↑ | ↑↑PCIe-MI Command↑ |
| | ↑↑Others↑ | ↑↑Reserved↑ |
| ↑↑13↑ | ↑↑Reserved↑ | |
| ↑↑12:8↑ | ↑↑Instance Identifier (Instance ID)↑ | ↑↑This field must indicate the value in the Instance Identifier field of the corresponding PCIe-MI Command .↑ |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|---|--|---|
| | ↑↑Bits↑ | ↑↑Description↑ |
| | ↑↑7↑ | ↑↑Integrity Check (IC)↑ : ↑↑For PCIe-MI over MCTP, this bit is Clear per the [PCIe-MI over MCTP Binding Specification – https://www.dmtf.org/dsp/DSP0291]. For PCIe-MI over MMPT, this bit is not applicable and must be Clear.↑ |
| | ↑↑6:0↑ | ↑↑Message Type↑ : ↑↑For PCIe-MI over MCTP, this field is set to 9 to indicate a PCIe-MI Message per the [PCIe-MI over MCTP Binding Specification - https://www.dmtf.org/dsp/DSP0291]. For PCIe-MI over MMPT, this field must be set to 9 to indicate a PCIe-MI Message (see the PCI Code and ID Assignment Specification).↑ |
| ↑↑2↑ | <p>↑↑PCIe-MI Message Header Version (PMHV)↑ : ↑↑This field indicates the version of the PCIe-MI Message Header. This field must be cleared to 0 for this version of the specification.↑</p> <p>↑↑See § Section 12.7.3 for additional details.↑</p> | |
| ↑↑3↑ | <p>↑↑PCIe-MI Message Header Length (PMHL)↑ : ↑↑This field indicates the length of the PCIe-MI Message Header in bytes. This field must be set to 8 for this version of the specification.↑</p> | |
| ↑↑4↑ | <p>↑↑Status:↑ ↑↑This field must indicate the status (see § Table 12-28) of the corresponding PCIe-MI Command.↑</p> | |
| ↑↑7:5↑ | <p>↑↑Reserved↑</p> | |
| <p>↑↑PCIe-MI Completion Content↑</p> | | |
| ↑↑N+7:8↑ | <p>↑↑This optional field indicates PCIe-MI Completion -specific content as defined in the following sections, where N is the size of this field in bytes. If N is 0, then this field is not present.↑</p> | |

12.7.2.1 ↑↑PCIe-MI Opcodes↑ §

 ECN: Base 6.3
 PCIe-MI △ ◀▷

↑↑PCIe-MI Commands↑ are ↑↓allowed, ↑↑identified by an opcode specified in the Opcode field of the PCIe-MI Command . The opcodes defined for PCIe-MI are listed in § Table 12-27 .↑

Table 12-27 ↑↑PCIe-MI Opcodes↑ §

| ↑↑Opcode↑ | ↑↑Description↑ | ↑↑Support↑ ↑↑1↑ | ↑↑Section↑ |
|---------------|--------------------|-----------------|-----------------------|
| ↑↑00h↑ | ↑↑Get Information↑ | ↑↑M↑ | ↑↑§ Section 12.7.3.1↑ |
| ↑↑01h↑ | ↑↑Get Parameters↑ | ↑↑O↑ ↑↑2↑ | ↑↑§ Section 12.7.3.2↑ |
| ↑↑02h↑ | ↑↑Set Parameters↑ | ↑↑O↑ ↑↑2↑ | ↑↑§ Section 12.7.3.2↑ |
| ↑↑03h to BFh↑ | ↑↑Reserved↑ | | |
| ↑↑C0h to FFh↑ | ↑↑Vendor Specific↑ | | |

↑↑Notes:↑

1. ↑↑O/M definition: O = optional; M = mandatory over MCTP for each physical layer binding over which PCIe-MI over MCTP is supported and mandatory over MMPT if PCIe-MI over MMPT is supported.↑
2. ↑↑If the PCIe-MI Completer does not support any Parameter Identifiers, then the Get Parameters command and Set Parameters command must not be implemented on that PCIe-MI Completer .↑

12.7.2.2 *PCIe-MI Status* §ECN: Base 6.3
PCIe-MI △

↑↑↑The status of a PCIe-MI Command must be returned in the Status field of the corresponding PCIe-MI Completion . The status codes defined for PCIe-MI are listed in § Table 12-28 . If the PCIe-MI Completer encounters an error corresponding to a status code in the range 10h to BFh while processing the PCIe-MI Command , then the PCIe-MI Command must be aborted prior to transmitting the PCIe-MI Completion .↑

IMPLEMENTATION NOTE: PCIE-MI COMMAND ERROR HANDLING §

↑↑↑If a PCIe-MI Commander receives a PCIe-MI Completion with the Status field set to a value in the range 10h to BFh which indicates the PCIe-MI Command was aborted, then the PCIe-MI Commander may choose to reissue the PCIe-MI Command in an attempt to recover from transient errors. If retrying the PCIe-MI Command does not resolve the issue and a PCIe-MI Completer Reset is able to be performed without disturbing the operation of Functions on the PCIe-MI Component that are in use by the operating system running on the host CPU(s), then a PCIe-MI Commander may choose to reset the PCIe-MI Completer at runtime using that PCIe-MI Completer Reset and then reissue the PCIe-MI Command again.↑

↑↑↑For example, a PCIe-MI Completer Reset of an MCTP endpoint on a 2-Wire interface port does not impact Functions on the PCIe-MI Component in use by the operating system running on the host CPU(s), whereas a PCIe-MI Completer Reset of a PCIe VDM MCTP endpoint may impact Functions on the PCIe-MI Component in use by the operating system running on the host CPU(s). If a PCIe-MI Completer Reset is not able to be performed without disturbing the operation of Functions on the PCIe-MI Component in use by the operating system running on the host CPU(s), then a reset or power cycle of the system the PCIe-MI Completer is installed in may be performed to attempt recovery.↑

↑↑↑If a PCIe-MI Commander receives a PCIe-MI Completion with the Status field set to 11h (i.e., External Error), then the error may be caused by an entity external to the PCIe-MI Component whose PCIe-MI Completer is processing the PCIe-MI Command . In such ↑↑↑cases, recovery of the error condition may require performing recovery steps on the external entity rather than the PCIe-MI Completer processing the PCIe-MI Command .↑

↑↑↑Unless otherwise specified, if multiple status codes apply, then the PCIe-MI Completer selects one of the applicable status codes to return in the Status field of the PCIe-MI Completion .↑

Table 12-28 *PCIe-MI Status Codes* §

| <i>Status Code</i> | <i>Description</i> |
|--|--|
| ↑↑↑Status Codes Indicating PCIe-MI Command Not Aborted↑ | |
| ↑↑↑00h↑ | ↑↑↑Success↑ : ↑↑↑The PCIe-MI Command was processed successfully.↑ |
| ↑↑↑01h to 0Fh↑ | ↑↑↑Reserved↑ |
| ↑↑↑Status Codes Indicating PCIe-MI Command Aborted↑ | |
| ↑↑↑10h↑ | ↑↑↑Internal Error↑ : ↑↑↑The PCIe-MI Command could not be processed due to a vendor-specific error internal to the PCIe-MI Component .↑ |
| ↑↑↑11h↑ | ↑↑↑External Error↑ : ↑↑↑The PCIe-MI Command could not be processed due to an error external to the PCIe-MI Component (e.g., a Configuration Write Request to a Function external to the PCIe-MI Component that is generated by |

Base 6.4 vs Base 6.3

| ↑↓Status Code↑ | ↑↓Description↑ |
|--|--|
| | the Set Parameters command specifying the Routing Information Parameter Identifier timed out or was completed with a UR or CA Completion Status).↑ |
| ↑↓12h↑ | ↑↓Invalid Input↑ : ↑↓The PCIe-MI Command contained a reserved, unimplemented, or conflicting value in a defined field (see § Section 12.7.2.2.1). This status code must only be used if there are no other status codes defined for the specific condition.↑ |
| ↑↓13h↑ | ↑↓PCIe-MI Message Content Length Error↑ : ↑↓The PCIe-MI Message Content contained greater than or less than the number of bytes required. See § Section 12.7.3 for additional details.↑ |
| ↑↓14h to BFh↑ | ↑↓Reserved↑ |
| ↑↓Vendor-Specific PCIe-MI Status Codes↑ | |
| ↑↓C0h to FFh↑ | ↑↓Vendor Specific↑ |

12.7.2.2.1 ↑↓Invalid Input Error Completion↑ §

 ECN: Base 6.3
 PCIe-MI △

↑↓An invalid input error occurs when a PCIe-MI Completer detects a reserved, unimplemented, or conflicting value in a defined bit or field in a PCIe-MI Command. An Invalid Input Error Completion is a PCIe-MI Completion with the Status field set to 12h (i.e., Invalid Input) and an Error Location field that indicates the bit or field that is invalid. The format of an Invalid Input Error Completion is shown in § Table 12-29.↑

↑↓Unless otherwise specified, if there are errors in multiple bits and/or fields in a PCIe-MI Command, then the PCIe-MI Completer selects which bit or field to indicate in the Error Location field.↑

Table 12-29 ↑↓Invalid Input Error Completion↑ §

| ↑↓Byte Location↑ | ↑↓Field Description↑ | |
|--|---|--|
| ↑↓7:0↑ | ↑↓PCIe-MI Message Header↑ : ↑↓This field must be set↑ as ↑↓per § Table 12-24 with the Status field set to 12h (i.e., Invalid Input).↑ | |
| ↑↓Invalid Input Error Completion Content↑ | | |
| | ↑↓Error Location↑ : ↑↓This field indicates the bit or field within the PCIe-MI Command that is invalid.↑ | |
| | ↑↓Bits↑ | ↑↓Description↑ |
| ↑↓9:8↑ | ↑↓15:3↑ | ↑↓Byte Location↑ : ↑↓This field must indicate the byte offset of the least-significant byte in the PCIe-MI Command of the bit or field that is invalid. If the byte offset of the invalid bit or field is beyond byte offset 8,191, then the value 8,191 must be indicated by this field.↑ |
| | ↑↓2:0↑ | ↑↓Bit Location↑ : ↑↓This field must indicate the bit offset of the least significant bit in the least significant byte in the PCIe-MI Command of the bit or field that is invalid.↑ |

12.7.3 ↑↑PCIe-MI Commands↑ §

ECN: Base 6.3
PCIe-MI△↔

↑↑This section defines the PCIe-MI Commands , which are identified by an Opcode (see § Table 12-27), and their corresponding PCIe-MI Completions . Each PCIe-MI Completer in a ↑↑USB↑↑PCIe-MI Component may support a different set of PCIe-MI Commands .↑

↑↑The following PCIe-MI Command and PCIe-MI Completion processing rules apply to ↑↑UART↑↑PCIe-MI Completers .↑

- ↑↑Reserved bits and fields in PCIe-MI Completions must be cleared to 0.↑
- ↑↑Reserved bits and fields in PCIe-MI Commands must be ignored.↑
- ↑↑A PCIe-MI Component must support the concurrent processing of PCIe-MI Commands on each PCIe-MI Completer on the PCIe-MI Component . The maximum number of supported concurrent PCIe-MI Commands on each PCIe-MI Completer is indicated by the Maximum Concurrent Commands fields (see § Table 12-39).↑
 - ↑↑As an example, if the PCIe-MI Component supports three PCIe-MI Completers (e.g., MCTP over PCIe VDM, MCTP over SMBus/I2C, and MMPT) and the Maximum Concurrent Commands field for each PCIe-MI Completer indicates a value of 2, then the PCIe-MI Component must support the concurrent processing of two PCIe-MI Commands on each of the three PCIe-MI Completers (i.e., concurrent processing of six PCIe-MI Commands in total).↑
- ↑↑If a PCIe-MI Completer receives a PCIe-MI Command (i.e., a new PCIe-MI Command) with an Instance Identifier equal to the Instance Identifier of a PCIe-MI Command that is currently being processed, then the PCIe-MI Command currently being processed may be aborted.↑
 - ↑↑If the PCIe-MI Command currently being processed is aborted, then a PCIe-MI Completion for the aborted PCIe-MI Command must not be returned.↑
 - ↑↑If the PCIe-MI Command currently being processed is not aborted, then an Invalid Input Error Completion with the Error Location field indicating the Instance ID field must be returned for the new PCIe-MI Command (see § Section 12.7.2.2.1) and the PCIe-MI Completer must continue processing the PCIe-MI Command with that Instance ID that is currently being processed.↑
- ↑↑If a PCIe-MI Completer receives a PCIe-MI Command (i.e., a new PCIe-MI Command) with an Instance Identifier that is not equal to the Instance Identifier of a PCIe-MI Command that is currently being processed while the PCIe-MI Completer is currently processing the maximum number of concurrent PCIe-MI Commands supported (see the Maximum Concurrent Commands field in § Table 12-39), then an Invalid Input Error Completion with the Error Location field indicating the Instance ID field must be returned for the new PCIe-MI Command and the PCIe-MI Completer must continue processing the PCIe-MI Command (s) currently being processed.↑

↑↑A PCIe-MI Completer designed to support version N of any PCIe-MI Command structure with a version field (e.g., the PCIe-MI Message Header Version field) behaves as follows.↑

- ↑↑Must return a value of N in the version field in the PCIe-MI Completion , if the version field is returned in the PCIe-MI Completion , regardless of the value of the version field in the corresponding PCIe-MI Command .↑
- ↑↑Must not generate an error for any value in the version field of the PCIe-MI Command .↑
- ↑↑Must process the structure as defined by version N of the structure regardless of the value in the version field of the structure, unless otherwise specified.↑
 - ↑↑If the value of the version field is less than the value the PCIe-MI Completer is compliant to, then the following rules apply.↑
 - ↑↑If the value of any associated length field (e.g., the PCIe-MI Message Header Length field) is less than the length defined by the initial version of the structure, then an Invalid

Base 6.4 vs Base 6.3

- Input Error Completion with the Error Location field indicating the length field must be returned.**
- **If the value of any associated length field is greater than the length defined by the initial version of the structure but not equal to the length defined for the specified version of the structure, then an Invalid Input Error Completion with the Error Location field indicating the length field should be returned.**
 - **If the value of any associated length field (e.g., the PCIe-MI Message Header Length field) is greater than the length defined by the version of the structure the PCIe-MI Completer is compliant to, then an Invalid Input Error Completion with the Error Location field indicating the length field must be returned.**
 - **If the value of any associated length field is greater than or equal to the length defined by the initial version of the PCIe-MI Message Header but less than the value defined by the revision of this specification that the PCIe-MI Completer is compliant to, then the structure must be processed without utilizing the information in structure bytes defined in a revision of this specification after the revision of this specification that the structure is compliant to.**
 - **If the value of the version field is greater than the value the PCIe-MI Completer is compliant to, then the following rules apply.**
 - **If the value of any associated length field is less than the length defined by the revision of this specification that the PCIe-MI Completer is compliant to, then an Invalid Input Error Completion with the Error Location field indicating the length field must be returned.**
 - **If the value of any associated length field is greater than the length defined by the revision of this specification that the PCIe-MI Completer is compliant to, then the extra bytes that are not defined in the revision of this specification that the PCIe-MI Completer is compliant to must be ignored.**
 - **Must not perform any functionality related to the structure that is not defined by version N of that structure.**
- A PCIe-MI Completer designed to support revision N of this specification handles the PCIe-MI Message Content as follows.**
- **Must not generate an error due to PCIe-MI Message Content being compliant to a revision of this specification other than the revision of this specification that the PCIe-MI Completer is compliant to.**
 - **Must process the PCIe-MI Message Content as defined by revision N of this specification regardless of the revision of this specification that the PCIe-MI Command is compliant to, if no errors with any length field in the PCIe-MI Command are detected, unless otherwise specified.**
 - **If the length of the PCIe-MI Message Content is less than the length defined by the initial version of the PCIe-MI Message Content, then a PCIe-MI Completion with the Status field set to a value of 13h (i.e., PCIe-MI Message Content Length Error) must be returned.**
 - **If the length of the PCIe-MI Message Content is greater than the length defined by the initial version of the PCIe-MI Message Content, less than the length defined by the revision of this specification the PCIe-MI Completer is compliant to, and not equal to the length defined by any version of the PCIe-MI Message Content that is less than the version the PCIe-MI Completer is compliant to, then a PCIe-MI Completion with the Status field set to a value of 13h (i.e., PCIe-MI Message Content Length Error) should be returned.**
 - **If the length of the PCIe-MI Message Content is greater than or equal to the length defined by the initial version of the PCIe-MI Message Content but less than the length defined by the revision of this specification that the PCIe-MI Completer is compliant to, then the PCIe-MI Message Content must be processed without utilizing the information in the PCIe-MI Message Content bytes defined in a**

revision of this specification after the revision of this specification that the PCIe-MI Command is compliant to.[↑]

- ^{↑↓}If the length of the PCIe-MI Message Content is greater than the length defined by the revision of this specification that the PCIe-MI Completer is compliant to, then the extra bytes that are not defined in the revision of this specification that the PCIe-MI Completer is compliant to must be ignored.[↑]
- ^{↑↓}Must not perform any functionality related to the PCIe-MI Message Content that is not defined the revision of this specification that the PCIe-MI Completer is compliant to.[↑]

^{↑↓}To ensure compatibility with PCIe-MI Completers that comply to any revision of this specification, a PCIe-MI Commander should process PCIe-MI Completions, including accounting for PCIe-MI Completions of varying sizes and versions, using techniques similar to the ones[↑] described ^{↑↓within} ^{↑↓in} the ^{↑↓}FRU attributes.^{↑↓} section for PCIe-MI Completers processing PCIe-MI Commands.[↑]

12.7.3.1 ^{↑↓}Get Information (Opcode 00h)[↑] §

ECN: Base 6.3
PCIe-MI△◀▷

^{↑↓}Definition[↑] ^{↑↓}This PCIe-MI Command returns information about a PCIe-MI Component. The type[↑] of ^{↑↓}specific targeted control functions[↑] ^{↑↓}information returned in the PCIe-MI Completion is specified by an Information Identifier in the Get Information command. The Information Identifiers supported by the Get Information command are listed in § Table 12-31. Each PCIe-MI Completer in a PCIe-MI Component may support a different set of Information Identifiers.[↑]

Table 12-31 ^{↑↓}Information Identifiers[↑] §

| ^{↑↓} Information Identifier [↑] | ^{↑↓} Description [↑] | ^{↑↓} Support [↑] ^{↑↓1↑} | ^{↑↓} Section [↑] |
|---|--|--|---|
| ^{↑↓} 00h [↑] | ^{↑↓} Supported Information Identifiers [↑] | ^{↑↓} M [↑] | ^{↑↓} § Section 12.7.3.1.1 [↑] |
| ^{↑↓} 01h [↑] | ^{↑↓} Supported Opcodes [↑] | ^{↑↓} M [↑] | ^{↑↓} § Section 12.7.3.1.2 [↑] |
| ^{↑↓} 02h [↑] | ^{↑↓} Supported Parameters [↑] | ^{↑↓} O/M [↑] ^{↑↓2↑} | ^{↑↓} § Section 12.7.3.1.3 [↑] |
| ^{↑↓} 03h [↑] | ^{↑↓} Component Information [↑] | ^{↑↓} O [↑] | ^{↑↓} § Section 12.7.3.1.4 [↑] |
| ^{↑↓} 04h [↑] | ^{↑↓} Port Information [↑] | ^{↑↓} O [↑] | ^{↑↓} § Section 12.7.3.1.5 [↑] |
| ^{↑↓} 05h [↑] | ^{↑↓} Function List [↑] | ^{↑↓} O [↑] | ^{↑↓} § Section 12.7.3.1.6 [↑] |
| ^{↑↓} 06h [↑] | ^{↑↓} Function Information [↑] | ^{↑↓} O [↑] | ^{↑↓} § Section 12.7.3.1.7 [↑] |
| ^{↑↓} 07h to BFh [↑] | ^{↑↓} Reserved [↑] | | |
| ^{↑↓} C0h to FFh [↑] | ^{↑↓} Vendor Specific [↑] | | |

^{↑↓}Notes[↑]:

1. ^{↑↓}O/M definition: O = optional; M = mandatory over MCTP for each physical layer binding over which PCIe-MI over MCTP is supported and mandatory over MMPT if PCIe-MI over MMPT is supported.[↑]
2. ^{↑↓}Mandatory if any optional Parameter Identifiers[↑] are ^{↑↓}supported by[↑] the ^{↑↓}responsibility[↑] ^{↑↓}PCIe-MI Completer; otherwise, the Supported Parameters Information Identifier must not be implemented by the PCIe-MI Completer.[↑]

^{↑↓}The PCIe-MI Command format for the Get Information command is shown in § Table 12-32.[↑]

Base 6.4 vs Base 6.3

Table 12-32 $\uparrow\downarrow$ Get Information Command $\uparrow\downarrow$ §

| $\uparrow\downarrow$ Byte Location $\uparrow\downarrow$ | $\uparrow\downarrow$ Field Description $\uparrow\downarrow$ |
|--|---|
| $\uparrow\downarrow$ 7:0 $\uparrow\downarrow$ | $\uparrow\downarrow$ PCIe-MI Message Header $\uparrow\downarrow$: $\uparrow\downarrow$ This field specifies values as per § Table 12-21 with the Opcode field cleared to 00h (i.e., Get Information, see § Table 12-27). $\uparrow\downarrow$ |
| $\uparrow\downarrow$Get Information Command Content$\uparrow\downarrow$ | |
| $\uparrow\downarrow$ 8 $\uparrow\downarrow$ | $\uparrow\downarrow$ Information Identifier $\uparrow\downarrow$: $\uparrow\downarrow$ This field specifies the Information Identifier (see § Table 12-31). If this field specifies an unsupported Information Identifier, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned. $\uparrow\downarrow$ |
| $\uparrow\downarrow$Information Identifier-Specific Command Content$\uparrow\downarrow$ | |
| $\uparrow\downarrow$ N+8:9 $\uparrow\downarrow$ | $\uparrow\downarrow$ Information Identifier-Specific Command Content $\uparrow\downarrow$: $\uparrow\downarrow$ This optional field specifies Information Identifier-specific command content as defined in the following subsections, where N is the size \uparrow of \downarrow form factor specifications. There \downarrow $\uparrow\downarrow$ this field in bytes. If N is 0, then this field is not present. $\uparrow\downarrow$ |
| $\uparrow\downarrow$ The PCIe-MI Completion format for the Get Information command for each Information Identifier is defined in the following subsections. $\uparrow\downarrow$ | |

12.7.3.1.1 $\uparrow\downarrow$ Supported Information Identifiers (Information Identifier 00h) $\uparrow\downarrow$ § ECN: Base 6.3
PCIe-MI Δ \triangleleft

$\uparrow\downarrow$ This command must return the list of optional and mandatory Information Identifiers (see § Table 12-31) in ascending order of Information Identifier supported by the PCIe-MI Completer on which the Get Information command is received. This command uses the PCIe-MI Command format shown in § Table 12-32 with the Information Identifier field cleared to 00h (i.e., Supported Information Identifiers, see § Table 12-31) and with no optional Information Identifier-specific command content. $\uparrow\downarrow$

$\uparrow\downarrow$ The PCIe-MI Completion format for this command is shown in § Table 12-33. $\uparrow\downarrow$

Table 12-33 $\uparrow\downarrow$ Get Information – Supported Information Identifiers Completion $\uparrow\downarrow$ §

| $\uparrow\downarrow$ Byte Location $\uparrow\downarrow$ | $\uparrow\downarrow$ Field Description $\uparrow\downarrow$ |
|---|---|
| $\uparrow\downarrow$ 7:0 $\uparrow\downarrow$ | $\uparrow\downarrow$ PCIe-MI Message Header $\uparrow\downarrow$: $\uparrow\downarrow$ This field must be set as per § Table 12-24. $\uparrow\downarrow$ |
| $\uparrow\downarrow$Supported Information Identifiers Completion Content$\uparrow\downarrow$ | |
| $\uparrow\downarrow$ 8 $\uparrow\downarrow$ | $\uparrow\downarrow$ Number of Information Identifiers $\uparrow\downarrow$: $\uparrow\downarrow$ This field must indicate the number of Information Identifiers returned (see § Table 12-31). $\uparrow\downarrow$ |
| $\uparrow\downarrow$ 9 $\uparrow\downarrow$ | $\uparrow\downarrow$ Information Identifier 0 $\uparrow\downarrow$: $\uparrow\downarrow$ This field must indicate the supported Information Identifier with the lowest value. $\uparrow\downarrow$ |
| $\uparrow\downarrow$ 10 $\uparrow\downarrow$ | $\uparrow\downarrow$ Information Identifier 1 $\uparrow\downarrow$: $\uparrow\downarrow$ This field must indicate the supported Information Identifier with the second lowest value, if there is more than one supported Information Identifier. $\uparrow\downarrow$ |
| $\uparrow\downarrow$... $\uparrow\downarrow$ | $\uparrow\downarrow$... $\uparrow\downarrow$ |
| $\uparrow\downarrow$ N+9 $\uparrow\downarrow$ | $\uparrow\downarrow$ Information Identifier N $\uparrow\downarrow$: $\uparrow\downarrow$ This field must indicate the supported Information Identifier with the highest value, if there \downarrow are $\uparrow\downarrow$ more than \downarrow two $\uparrow\downarrow$ types \downarrow $\uparrow\downarrow$ supported Information Identifiers, where N is the number \uparrow of \downarrow control targets, \downarrow $\uparrow\downarrow$ Information Identifiers in the list minus 1. $\uparrow\downarrow$ |

Base 6.4 vs Base 6.3

12.7.3.1.2 $\uparrow\downarrow$ Supported Opcodes (Information Identifier 01h)ECN: Base 6.3
PCIe-MI Δ $\triangleleft\triangleright$

$\uparrow\downarrow$ This command must return the list of optional and mandatory opcodes (see § Table 12-27) in ascending order of opcode supported by the PCIe-MI Completer on which the Get Information command is received. This command uses the PCIe-MI Command format shown in § Table 12-32 with the Information Identifier field set to 01h (i.e., Supported Opcodes, see § Table 12-31) and with no optional Information Identifier-specific command content. \uparrow

$\uparrow\downarrow$ The PCIe-MI Completion format for this command is shown in § Table 12-34. \uparrow

Table 12-34 $\uparrow\downarrow$ Get Information – Supported Opcodes Completion

| $\uparrow\downarrow$ Byte Location | $\uparrow\downarrow$ Field Description |
|---|---|
| $\uparrow\downarrow$ 7:0 \uparrow | $\uparrow\downarrow$ PCIe-MI Message Header : $\uparrow\downarrow$ This field must be set as per § Table 12-24. \uparrow |
| $\uparrow\downarrow$ Supported Opcodes Completion Content | |
| $\uparrow\downarrow$ 8 \uparrow | $\uparrow\downarrow$ Number of Opcodes : $\uparrow\downarrow$ This field must indicate the number of Opcodes returned (see § Table 12-27). \uparrow |
| $\uparrow\downarrow$ 9 \uparrow | $\uparrow\downarrow$ Opcode 0 : $\uparrow\downarrow$ This field must indicate the supported Opcode with the lowest value. \uparrow |
| $\uparrow\downarrow$ 10 \uparrow | $\uparrow\downarrow$ Opcode 1 : $\uparrow\downarrow$ This field must indicate the supported Opcode with the second lowest value, if there is more than one supported Opcode. \uparrow |
| $\uparrow\downarrow$... \uparrow | $\uparrow\downarrow$... \uparrow |
| $\uparrow\downarrow$ N+9 \uparrow | $\uparrow\downarrow$ Opcode N : $\uparrow\downarrow$ This field must indicate the Opcode with the highest value, if there are more than two supported Opcodes, where N is the number of Opcodes in the list minus 1. \uparrow |

12.7.3.1.3 $\uparrow\downarrow$ Supported Parameters (Information Identifier 02h)ECN: Base 6.3
PCIe-MI Δ $\triangleleft\triangleright$

$\uparrow\downarrow$ This command must return the list of Parameter Identifiers (see § Table 12-56) in ascending order of Parameter Identifier supported by the PCIe-MI Completer on which the Get Information command is received. If a Parameter Identifier is supported, then the Get Parameters command for that Parameter Identifier must be supported and the Set Parameters command for that Parameter Identifier is optional. This command uses the PCIe-MI Command format shown in § Table 12-32 with the Information Identifier field set to 02h (i.e., Supported Parameters, see § Table 12-31) and with no optional Information Identifier-specific command content. \uparrow

$\uparrow\downarrow$ The PCIe-MI Completion format for this command is shown in § Table 12-35. \uparrow

Table 12-35 $\uparrow\downarrow$ Get Information – Supported Parameters Completion

| $\uparrow\downarrow$ Byte Location | $\uparrow\downarrow$ Field Description |
|--|---|
| $\uparrow\downarrow$ 7:0 \uparrow | $\uparrow\downarrow$ PCIe-MI Message Header : $\uparrow\downarrow$ This field must be set as per § Table 12-24. \uparrow |
| $\uparrow\downarrow$ Supported Parameters Completion Content | |
| $\uparrow\downarrow$ 8 \uparrow | $\uparrow\downarrow$ Number of Parameters : $\uparrow\downarrow$ This field must indicate the number of Parameter Identifiers returned (see § Table 12-56). \uparrow |
| $\uparrow\downarrow$ 10:9 \uparrow | $\uparrow\downarrow$ Parameter 0 Information : $\uparrow\downarrow$ This field must indicate information about the supported Parameter Identifier with the lowest value. \uparrow |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|---------------------|----------------------|---|
| | ↑↑Bits↑ | ↑↑Description↑ |
| | ↑↑15:9↑ | ↑↑Reserved↑ |
| | ↑↑8↑ | ↑↑Set Parameters Support↑ : ↑↑If the Set Parameters command for this Parameter Identifier is supported, then this bit must be Set; otherwise, this bit must be Clear.↑ |
| | ↑↑7:0↑ | ↑↑Parameter 0↑ : ↑↑This field must indicate the Parameter Identifier with the lowest value that is supported by the Get Parameters command.↑ |
| | | ↑↑Parameter 1 Information↑ : ↑↑This field must indicate information about the supported Parameter Identifier with the second lowest value, if there is more than one supported Parameter Identifier.↑ |
| ↑↑12:11↑ | ↑↑Bits↑ | ↑↑Description↑ |
| | ↑↑15:9↑ | ↑↑Reserved↑ |
| | ↑↑8↑ | ↑↑Set Parameters Support↑ : ↑↑If the Set Parameters command for this Parameter Identifier is supported, then this bit must be Set; otherwise, this bit must be Clear.↑ |
| | ↑↑7:0↑ | ↑↑Parameter 1↑ : ↑↑This field must indicate the Parameter Identifier with the second lowest value that is supported by the Get Parameters command.↑ |
| ↑↑...↑ | ↑↑...↑ | ↑↑Parameter N Information↑ : ↑↑This field must indicate information about the supported Parameter Identifier with the highest value, if there are more than two supported Parameter Identifiers,↑ where ↑↑N↑ is the number of Parameter Identifiers in the list minus 1.↑ |
| ↑↑(N*2)+10:(N*2)+9↑ | ↑↑Bits↑ | ↑↑Description↑ |
| | ↑↑15:9↑ | ↑↑Reserved↑ |
| | ↑↑8↑ | ↑↑Set Parameters Support↑ : ↑↑If the Set Parameters command for this Parameter Identifier is supported, then this bit must be Set; otherwise, this bit must be Clear.↑ |
| | ↑↑7:0↑ | ↑↑Parameter↑ ↑↑N: This field must indicate the Parameter Identifier with the highest value that is supported by the Get Parameters command.↑ |

12.7.3.1.4 ↑↑Component Information (Information Identifier 03h)↑

ECN: Base 6.3
PCIe-MI△◀▷

↑↑This command returns information about the PCIe-MI Component . This command uses the PCIe-MI Command format shown in § Table 12-32 with the Information Identifier field set to 03h (i.e., Component Information, see § Table 12-31) and with no optional Information Identifier-specific command content.↑

↑↑The PCIe-MI Completion format for this command is shown in § Table 12-39 .↑

Table 12-39 ↑↑Get Information – Component Information Completion↑

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|------------------|---------------------------|--|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ | ↑↑This field must be set as per § Table 12-24 .↑ |

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|---|--|
| ↑↑Component Information Completion Content↑ | |
| ↑↑8↑ | ↑↑Number of Ports↑ : ↑↑This field must indicate the number of supported PCIe-MI Ports on the PCIe-MI Component .↑ |
| ↑↑9↑ | ↑↑Maximum Concurrent Commands↑ : ↑↑This field must indicate the maximum number of PCIe-MI Commands that are able to be processed concurrently by the PCIe-MI Completer that is generating this PCIe-MI Completion . This field must be set to a value in the range of 1 to 32 and all other values are reserved.↑ |
| ↑↑11:10↑ | ↑↑Maximum Command-To-Completion Time↑ : ↑↑This field must indicate the estimated maximum time in 100 ms units required by the PCIe-MI Completer that is generating this PCIe-MI Completion to process a PCIe-MI Command and begin transmission of the PCIe-MI Completion , assuming the physical transport external to the PCIe-MI Completer is available. This time is measured by the PCIe-MI Completer from the end of the reception of a PCIe-MI Command to the beginning of the transmission of the corresponding PCIe-MI Completion .↑ |

12.7.3.1.5 ↑↑Port Information (Information Identifier 04h)↑

 ECN: Base 6.3
 PCIe-MI△◀▷

↑↑This command returns information about the PCIe-MI Port specified by the PCIe-MI Port Identifier field. This command uses the PCIe-MI Command format shown in § Table 12-40 .↑

Table 12-40 ↑↑Get Information – Port Information Command↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|--|---|
| ↑↑Port Information Command Content↑ | |
| ↑↑8↑ | ↑↑Information Identifier↑ : ↑↑This field specifies a value of 04h (i.e., Port Information, see § Table 12-31).↑ |
| ↑↑Port Information-Specific Command Content↑ | |
| ↑↑9↑ | ↑↑PCIe-MI Port Identifier↑ : ↑↑This field specifies the PCIe-MI Port Identifier. If this field specifies a PCIe-MI Port Identifier that does not exist, then an ↑↑Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |

↑↑The PCIe-MI Completion format for this command is shown in § Table 12-24 .↑

Table 12-41 ↑↑Get Information – Port Information Completion↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|--|--|
| ↑↑Port Information Completion Content↑ | |
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field must be set as per § Table 12-24 .↑ |
| ↑↑8↑ | ↑↑Port Description↑ : ↑↑This field indicates the type of PCIe-MI Port and whether the PCIe-MI Port is exposed to the host system.↑ |
| ↑↑7↑ | ↑↑PCIe-MI Port Exposed↑ : If this PCIe-MI Port is exposed to the host system (see § Section 12.7.1), then this bit must be Set; otherwise, this bit must be Clear.↑ |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---------------------------------------|--|----------|----------------|---------------------------------------|---------|-----------------------|----------------------------|--------|---|----------------------------|--------|---|-------|--------|---|-------|------|---|--|------|---|--|
| ↑↑Bits↑ | ↑↑Description↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑16:2↑ | ↑↑Reserved↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑11:0↑ | <p>↑↑PCIe-MI Port Type↑ : ↑↑This field must indicate the type of this PCIe-MI Port.↑</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">↑↑Value↑</th> <th style="text-align: center;">↑↑Description↑</th> <th style="text-align: center;">↑↑Port-Specific Information Required↑</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">↑↑00b↑</td><td style="text-align: center;">↑↑PCIe Upstream Port↑</td><td style="text-align: center;">↑↑Yes (see § Table 12-44)↑</td></tr> <tr> <td style="text-align: center;">↑↑01b↑</td><td style="text-align: center;">↑↑PCIe Downstream Port↑</td><td style="text-align: center;">↑↑Yes (see § Table 12-44)↑</td></tr> <tr> <td style="text-align: center;">↑↑10b↑</td><td style="text-align: center;">↑↑2-Wire Interface Port↑</td><td style="text-align: center;">↑↑No↑</td></tr> <tr> <td style="text-align: center;">↑↑11b↑</td><td style="text-align: center;">↑↑USB Port↑</td><td style="text-align: center;">↑↑No↑</td></tr> </tbody> </table> | | | ↑↑Value↑ | ↑↑Description↑ | ↑↑Port-Specific Information Required↑ | ↑↑00b↑ | ↑↑PCIe Upstream Port↑ | ↑↑Yes (see § Table 12-44)↑ | ↑↑01b↑ | ↑↑PCIe Downstream Port↑ | ↑↑Yes (see § Table 12-44)↑ | ↑↑10b↑ | ↑↑2-Wire Interface Port↑ | ↑↑No↑ | ↑↑11b↑ | ↑↑USB Port↑ | ↑↑No↑ | | | | | | |
| ↑↑Value↑ | ↑↑Description↑ | ↑↑Port-Specific Information Required↑ | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑00b↑ | ↑↑PCIe Upstream Port↑ | ↑↑Yes (see § Table 12-44)↑ | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑01b↑ | ↑↑PCIe Downstream Port↑ | ↑↑Yes (see § Table 12-44)↑ | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑10b↑ | ↑↑2-Wire Interface Port↑ | ↑↑No↑ | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑11b↑ | ↑↑USB Port↑ | ↑↑No↑ | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑11:9↑ | ↑↑Reserved↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑Port-Specific Information Completion Content↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑N:12↑ | <p>↑↑Port-Specific Information↑ : ↑↑This optional field contains port-specific completion content.↑</p> <p>↑↑The PCIe-MI Completion for this command that indicates a PCIe Upstream Port or PCIe Downstream Port is shown in § Table 12-44.↑</p> | | | | | | | | | | | | | | | | | | | | | | | |
| <i>Table 12-44 ↑↑Get Information – PCIe Port-Specific Information Completion↑</i> § | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑Byte Location↑ | ↑↑Field Description↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑11:0↑ | ↑↑This field must be set as per § Table 12-24.↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑PCIe Port-Specific Information Completion Content↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑13:12↑ | <p>↑↑Supported Link Speeds Vector↑ : ↑↑This field indicates the supported Link speeds of this PCIe Port.↑</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">↑↑Bits↑</th> <th colspan="2" style="text-align: center;">↑↑Description↑</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">↑↑15:6↑</td><td colspan="2" style="text-align: center;">↑↑Reserved↑</td></tr> <tr> <td style="text-align: center;">↑↑5↑</td><td colspan="2"> <p>↑↑64.0 GT/s Support↑ : ↑↑If the Link supports 64.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> </td></tr> <tr> <td style="text-align: center;">↑↑4↑</td><td colspan="2"> <p>↑↑32.0 GT/s Support↑ : ↑↑If the Link supports 32.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> </td></tr> <tr> <td style="text-align: center;">↑↑3↑</td><td colspan="2"> <p>↑↑16.0 GT/s Support↑ : ↑↑If the Link supports 16.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> </td></tr> <tr> <td style="text-align: center;">↑↑2↑</td><td colspan="2"> <p>↑↑8.0 GT/s Support↑ : ↑↑If the Link supports 8.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> </td></tr> <tr> <td style="text-align: center;">↑↑1↑</td><td colspan="2"> <p>↑↑5.0 GT/s Support↑ : ↑↑If the Link supports 5.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> </td></tr> </tbody> </table> | | | ↑↑Bits↑ | ↑↑Description↑ | | ↑↑15:6↑ | ↑↑Reserved↑ | | ↑↑5↑ | <p>↑↑64.0 GT/s Support↑ : ↑↑If the Link supports 64.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | ↑↑4↑ | <p>↑↑32.0 GT/s Support↑ : ↑↑If the Link supports 32.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | ↑↑3↑ | <p>↑↑16.0 GT/s Support↑ : ↑↑If the Link supports 16.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | ↑↑2↑ | <p>↑↑8.0 GT/s Support↑ : ↑↑If the Link supports 8.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | ↑↑1↑ | <p>↑↑5.0 GT/s Support↑ : ↑↑If the Link supports 5.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | |
| ↑↑Bits↑ | ↑↑Description↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑15:6↑ | ↑↑Reserved↑ | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑5↑ | <p>↑↑64.0 GT/s Support↑ : ↑↑If the Link supports 64.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑4↑ | <p>↑↑32.0 GT/s Support↑ : ↑↑If the Link supports 32.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑3↑ | <p>↑↑16.0 GT/s Support↑ : ↑↑If the Link supports 16.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑2↑ | <p>↑↑8.0 GT/s Support↑ : ↑↑If the Link supports 8.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↑1↑ | <p>↑↑5.0 GT/s Support↑ : ↑↑If the Link supports 5.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑</p> | | | | | | | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | | |
|------------------|---|---|--|
| | ↑↑Bits↑ | ↑↑Description↑ | |
| | ↑↑0↑ | ↑↑2.5 GT/s Support↑ : ↑↑If the Link supports 2.5 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| | | | |
| | ↑↑Current Link Speed↑ | ↑↑This field indicates information about the current Link speed of this PCIe port per the following table. If there is no Link (i.e., the PCIe-MI Port Exposed bit for the PCIe-MI Port is Clear) or the Link is not active (i.e., the status of the Data Link Control and Management State Machine is not DL_Active), then this field must be cleared to 0000h.↑ | |
| | ↑↑Bits↑ | ↑↑Description↑ | |
| ↑↑15:14↑ | ↑↑15:6↑ | ↑↑Reserved↑ | |
| | ↑↑5↑ | ↑↑64.0 GT/s Current Link Speed↑ : ↑↑If the current Link speed is 64.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| | ↑↑4↑ | ↑↑32.0 GT/s Current Link Speed↑ : ↑↑If the current Link speed is 32.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| | ↑↑3↑ | ↑↑16.0 GT/s Current Link Speed↑ : ↑↑If the current Link speed is 16.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| | ↑↑2↑ | ↑↑8.0 GT/s Current Link Speed↑ : ↑↑If the current Link speed is 8.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| | ↑↑1↑ | ↑↑5.0 GT/s Current Link Speed↑ : ↑↑If the current Link speed is 5.0 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| | ↑↑0↑ | ↑↑2.5 GT/s Current Link Speed↑ : ↑↑If the current Link speed is 2.5 GT/s, then this bit must be Set; otherwise, this bit must be Clear.↑ | |
| ↑↑16↑ | ↑↑Maximum Link Width↑ : ↑↑This field must indicate the supported Link width of this PCIe port. Unlike the Maximum Link Width field in the Link Capabilities Register , this value is not permitted to exceed the number of Lanes routed to the slot (Downstream Port), adapter connector (Upstream Port), or component in the case of component-to-component connections, the actual wired connection width. This field is not required to account for card carriers (see § Figure 12-12). Card carriers are able to advertise their maximum Link widths in the Connector Subdivision Combinations Descriptor (see § Table 12-20) in the card carrier's FRU Information Device (see § Table 12-12).↑ | | |
| ↑↑17↑ | ↑↑Negotiated Link Width↑ : ↑↑If the Link is active (i.e., the status of the Port's Data Link Control and Management State Machine is DL_Active), then this field must indicate the same value as the Negotiated Link Width field in the Port's Link Status Register . If there is no Link (i.e., the PCIe-MI Port Exposed bit for the PCIe-MI Port is Clear), or the Link is not active (i.e., the status of the Port's Data Link Control and Management State Machine is not DL_Active), then this field must be cleared to 0.↑ | | |
| ↑↑18↑ | ↑↑PCIe Port Number↑ : ↑↑This field must indicate the same value as the Port Number field in the Port's Link Capabilities Register .↑ | | |
| ↑↑19↑ | ↑↑Starting Lane Number↑ : ↑↑This field must indicate the lowest-numbered Lane of this PCIe port using the connector's designated lane numbering without lane reversal.↑ | | |
| ↑↑20↑ | ↑↑Ending Lane Number↑ : ↑↑This field must indicate the highest-numbered Lane of this PCIe port using the connector's designated lane numbering without lane reversal.↑ | | |

↑↑↑The PCIe-MI Completion for 2-Wire interface ports must be set as per § Table 12-24 with no Port-Specific Information field.↑

↑↑↑The PCIe-MI Completion for USB ports must be set as per § Table 12-24 with no Port-Specific Information field.↑

12.7.3.1.6 ↑↑Function List (Information Identifier 05h)↑

ECN: Base 6.3
PCIe-MI△
↔

↑↑↑This command returns a list of non IOV Functions and Physical Functions matching the specified command inputs that are implemented by the PCIe-MI Component . Virtual Functions must not be included in the PCIe-MI Completion . The PCIe-MI Completion for this command must return the following.↑

- ↑↑↑A Function List Data Structure for each non IOV Function and Physical Function implemented by the PCIe-MI Component , up to the number of Functions specified by the Maximum Function List Data Structures field, starting at the value specified by the Starting Function Identifier field.↑
- ↑↑↑A list of Function List Data Structures in ascending order of Function Identifier.↑
- ↑↑↑A list of Function List Data Structures whose Functions match the type specified by one of the bits Set in the Function Type field.↑

↑↑↑The Function Identifier for each Function in the PCIe-MI Component must be a four-byte value that is ↑↑↑unique within the PCIe-MI Component . Function Identifiers in a PCIe-MI Component are↑ permitted to ↑↑↑include↑ ↑↑↑be any four-byte value and are not required to be consecutive.↑

↑↑↑This command uses the PCIe-MI Command format shown in § Table 12-47 .↑

Table 12-47 ↑↑Get Information – Function List Command↑

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|--|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field specifies values as per § Table 12-21 with the Opcode field cleared to 00h.↑ |
| ↑↑Function List Command Content↑ | |
| ↑↑8↑ | ↑↑Information Identifier↑ : ↑↑This field specifies a ↑↑primary and/or secondary target. In↓ ↑↑value of 05h (i.e., Function List, see § Table 12-31).↑ |
| ↑↑Function List-Specific Command Content↑ | |
| ↑↑12:9↑ | ↑↑Starting Function Identifier↑ : ↑↑This field specifies the starting Function Identifier. If↑ this ↑↓context, target refers↑ ↑↑field specifies a Function Identifier that is greater than the maximum Function Identifier supported by the PCIe-MI Component , then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |
| ↑↑14:13↑ | ↑↑Maximum Function List Data Structures↑ : ↑↑This field specifies the maximum number of Function List Data Structures that are permitted↑ to ↑↑be returned.↑ |
| ↑↑16:15↑ | ↑↑Function Type↑ : ↑↑This field specifies a bitmask of Function types (referred to as↑ the ↑↑physical address↑ ↑↑Device/Port Type field in the PCI Express Capabilities Register). If a bit in this field is Set and the criteria specified by the Starting Function Identifier field↑ and ↑↑Maximum Function List Data Structures field are met for a given Function of the type specified by that bit, then that Function must be included in the PCIe-MI Completion ; otherwise, that Function must↑ not ↑↑be included in the PCIe-MI Completion .↑ |
| | ↑↑↑This field specifying a value of 0 is not an error.↑ |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|------------------|--|
| ↑↑Bits↑ | ↑↑Description↑ |
| ↑↑15:9↑ | ↑↑Reserved↑ |
| ↑↑8↑ | ↑↑PCI/PCI-X to PCI Express Bridge↑ |
| ↑↑7↑ | ↑↑PCI Express to PCI/PCI-X Bridge↑ |
| ↑↑6↑ | ↑↑Downstream Port of PCI Express Switch↑ |
| ↑↑5↑ | ↑↑Upstream Port of PCI Express Switch↑ |
| ↑↑4↑ | ↑↑Root Port of PCI Express Root Complex↑ |
| ↑↑3↑ | ↑↑Root Complex Event Collector↑ |
| ↑↑2↑ | ↑↑RCiEP↑ |
| ↑↑1↑ | ↑↑Legacy PCI Express Endpoint↑ |
| ↑↑0↑ | ↑↑PCI Express Endpoint↑ |

↑↑The PCIe-MI Completion format for this command is shown in § Table 12-49 .↑

Table 12-49 ↑↑Get Information – Function List Completion↑

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|--|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field must be set as per § Table 12-24 .↑ |
| ↑↑Function List Completion Header↑ | |
| ↑↑9:8↑ | ↑↑Number of Functions↑ : ↑↑This field must indicate the number of Function List Data Structures returned (see § Table 12-50).↑ |
| ↑↑10↑ | ↑↑Function List Version↑ : ↑↑This field indicates the version of the Function List Completion Header and Function List Completion Content. This field must be cleared to 0 for this version of the specification.↑ |
| ↑↑11↑ | ↑↑Function List Completion Header Length↑ : ↑↑This field indicates the length in bytes of the Function List Completion Header. This field must be set to 5 for this version of the specification.↑ |
| ↑↑12↑ | ↑↑Function List Data Structure Length↑ : ↑↑This field indicates the length in bytes of each Function List Data Structure (see § Table 12-50). This field must be set to 4 for this version of the specification.↑ |
| ↑↑Function List Completion Content↑ | |
| ↑↑M+12:13↑ | ↑↑Function List 0↑ : ↑↑This field must indicate the Function List Data Structure (see § Table 12-50) matching the specified command inputs of the Function Identifier with the lowest value, if there is at least one supported Function that matches the specified command inputs, where M is the value indicated by the Function List Data Structure Length field.↑ |
| ↑↑2*M+12:M+13↑ | ↑↑Function List 1↑ : ↑↑This field must indicate the Function List Data Structure of the Function Identifier with the second lowest value that matches the specified command inputs, if there are at least two supported Functions matching the specified command inputs.↑ |
| ↑↑...↑ | ↑↑...↑ |

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|----------------------|--|
| ↑↑(N+1)*M+12:N*M+13↑ | ↑↑Function List N↑ : ↑↑This field must indicate the Function List Data Structure of the Function Identifier with the highest value that matches the specified command inputs, if there are more than two Functions matching the specified command inputs, where N is the number of Function List Data Structures in the list minus 1.↑ |

↑↑The Function List Data Structure format is shown in § Table 12-50 .↑

Table 12-50 ↑↑Function List Data Structure↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|------------------|---|
| ↑↑3:0↑ | ↑↑Function Identifier↑ : ↑↑This field must indicate the Function Identifier of the Function.↑ |

12.7.3.1.7 ↑↑Function Information (Information Identifier 06h)↑ §

ECN: Base 6.3
PCIe-MI△<△

↑↑This command returns information about the Function specified by the Function Identifier field. This command uses the PCIe-MI Command format shown in § Table 12-51 .↑

Table 12-51 ↑↑Get Information – Function Information Command↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|---|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field specifies values as per § Table 12-21 with the Opcode field cleared to 00h.↑ |
| ↑↑Function Information Command Content↑ | |
| ↑↑8↑ | ↑↑Information Identifier↑ : ↑↑This field specifies a value of 06h (i.e., Function Information, see § Table 12-31).↑ |
| ↑↑Function Information-Specific Command Content↑ | |
| ↑↑12:9↑ | ↑↑Function Identifier↑ : ↑↑This field specifies the Function Identifier. If this field specifies a Function Identifier that exists in the PCIe-MI Component , then the information for the Function corresponding to that Function Identifier must be returned; otherwise, an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |

↑↑The PCIe-MI Completion format for this command is shown in § Table 12-52 .↑

Table 12-52 ↑↑Get Information – Function Information Completion↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|---|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field must be set as per § Table 12-24 .↑ |
| ↑↑Function Information Completion Content↑ | |
| ↑↑9:8↑ | ↑↑Vendor ID↑ : ↑↑This field must indicate the Vendor ID of the specified Function.↑ |
| ↑↑11:10↑ | ↑↑Device ID↑ : ↑↑This field must indicate the Device ID of the specified Function.↑ |
| ↑↑13:12↑ | ↑↑Subsystem Vendor ID↑ : ↑↑This field must indicate the Subsystem Vendor ID of the specified Function.↑ |
| ↑↑15:14↑ | ↑↑Subsystem ID↑ : ↑↑This field must indicate the Subsystem ID of the specified Function.↑ |
| ↑↑18:16↑ | ↑↑Class Code↑ : ↑↑This field must indicate the Class Code of the specified Function.↑ |
| ↑↑19↑ | ↑↑Function Description↑ : ↑↑This field describes the specified Function.↑ |

Base 6.4 vs Base 6.3

| ↑↓Byte Location↑ | ↑↓Field Description↑ | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------|---|---|----------|----------------|-------|-------------------------|-------|--------------------------------|-------|----------|-------|---------------------------------|-------|--|-------|--|-------|--|-------|------------------------------------|-------|------------------------------------|-----------|-------------|--|
| ↑↓Bits↑ | ↑↓Description↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓7↑ | ↑↓PCIe VDM↑ MCTP Endpoint↑ ↑↓IDs (EIDs) : ↑↓If the specified Function has a PCIe VDM MCTP endpoint (see the [MCTP PCIe VDM Binding Specification - https://www.dmtf.org/dsp/DSP0238]), then this bit must be Set; otherwise, this bit must be Clear.↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓6:4↑ | ↑↓Reserved↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓3:0↑ | ↑↓Function Type↑ : ↑↓This field must indicate the specified Function's type (referred to as the Device/Port Type field in the PCI Express Capabilities Register).↑ | <table border="1"> <thead> <tr> <th style="background-color: #ffffcc;">↑↓Value↑</th> <th style="background-color: #ffffcc;">↑↓Description↑</th> </tr> </thead> <tbody> <tr> <td style="background-color: #ffffcc;">↑↓0h↑</td><td>↑↓PCI Express Endpoint↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓1h↑</td><td>↑↓Legacy PCI Express Endpoint↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓2h↑</td><td>↑↓RCiEP↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓3h↑</td><td>↑↓Root Complex Event Collector↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓4h↑</td><td>↑↓Root Port of PCI Express Root Complex↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓5h↑</td><td>↑↓Upstream Port of PCI Express Switch↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓6h↑</td><td>↑↓Downstream Port of PCI Express Switch↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓7h↑</td><td>↑↓PCI Express to PCI/PCI-X Bridge↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓8h↑</td><td>↑↓PCI/PCI-X to PCI Express Bridge↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↓Others↑</td><td>↑↓Reserved↑</td></tr> </tbody> </table> | ↑↓Value↑ | ↑↓Description↑ | ↑↓0h↑ | ↑↓PCI Express Endpoint↑ | ↑↓1h↑ | ↑↓Legacy PCI Express Endpoint↑ | ↑↓2h↑ | ↑↓RCiEP↑ | ↑↓3h↑ | ↑↓Root Complex Event Collector↑ | ↑↓4h↑ | ↑↓Root Port of PCI Express Root Complex↑ | ↑↓5h↑ | ↑↓Upstream Port of PCI Express Switch↑ | ↑↓6h↑ | ↑↓Downstream Port of PCI Express Switch↑ | ↑↓7h↑ | ↑↓PCI Express to PCI/PCI-X Bridge↑ | ↑↓8h↑ | ↑↓PCI/PCI-X to PCI Express Bridge↑ | ↑↓Others↑ | ↑↓Reserved↑ | |
| ↑↓Value↑ | ↑↓Description↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓0h↑ | ↑↓PCI Express Endpoint↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓1h↑ | ↑↓Legacy PCI Express Endpoint↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓2h↑ | ↑↓RCiEP↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓3h↑ | ↑↓Root Complex Event Collector↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓4h↑ | ↑↓Root Port of PCI Express Root Complex↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓5h↑ | ↑↓Upstream Port of PCI Express Switch↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓6h↑ | ↑↓Downstream Port of PCI Express Switch↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓7h↑ | ↑↓PCI Express to PCI/PCI-X Bridge↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓8h↑ | ↑↓PCI/PCI-X to PCI Express Bridge↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓Others↑ | ↑↓Reserved↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓20↑ | ↑↓Number of Capabilities↑ : ↑↓This field must indicate the number of Capability Data Structures in the Capability List field.↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓21↑ | ↑↓Capability Data Structure Length↑ : ↑↓This field indicates the length in bytes of each Capability Data Structure in the Capability List field. This field must be set to 4 for this version of the specification.↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓N+21:22↑ | ↑↓Capability List↑ : ↑↓If the Configuration Space Parameter Identifier (i.e., Parameter Identifier 02h) is supported, then this field must indicate a list containing a Capability Data Structure (see § Table 12-55) for each Capability structure supported by the specified Function using the Get Parameters command with the Configuration Space Parameter Identifier, where N is the number of bytes in the list; otherwise, this field must indicate a list containing a Capability Data Structure for each Capability structure supported by the specified Function using a Configuration Read Request.↑ | | | | | | | | | | | | | | | | | | | | | | | | |
| ↑↓N+22↑ | ↑↓Path Origin↑ : ↑↓If the specified Function is downstream from an Upstream Port on the PCIe-MI Component, then this field must indicate the PCIe-MI Port Identifier of that Upstream Port.↑ ↑↓If the specified Function is not downstream from an Upstream Port on the PCIe-MI Component (i.e., a Function in a Root Complex) and the Path Length field does not specify a value of 0, then this field must contain a vendor-specific unique identifier that indicates which Segment the specified Function is on. If the specified Function is not downstream from an Upstream Port on the PCIe-MI Component and the Path Length field specifies a value of 0, then no Segment information is indicated by this field and this field must be cleared to 0.↑ | | | | | | | | | | | | | | | | | | | | | | | | |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|------------------|--|
| ↑↑N+23↑ | <p>↑↑Path Length↑ : ↑↑The Path field is optional but strongly recommended for Functions in Root Complexes; otherwise, the Path field is mandatory. If the Path field is supported, then this field must indicate the length of the Path field in bytes; otherwise, this field must be cleared to 0.↑</p> |
| ↑↑M+N+23:N+24↑ | <p>↑↑Path↑ : ↑↑If the value of the Path Length field does not specify a value of 0, then this field must indicate the hierarchical path on the PCIe-MI Component starting with the Function on the PCIe-MI Component that is furthest upstream from the specified Function and ending with the specified Function, where M is the number of bytes in this field (i.e., the value in the Path Length field) and N is the number of bytes in the Capability List field.↑</p> <p>↑↑Each byte in this field represents a Function along the path to the specified Function in order from the furthest upstream Function on the PCIe-MI Component in the lowest byte offset of this field to the specified Function in the highest byte offset of this field. Each byte contains the least-significant 8-bits of the Routing ID of the Function (e.g., the Device Number in bits 7:3 and the Function Number in bits 3:0 for non-ARI Functions, or the Function Number in bits 7:0 for ARI Functions).↑</p> <p>↑↑If there is only one Function in the path, then the path consists of a single byte representing the specified Function.↑</p> <p>↑↑If there is more than one Function in the path, then the byte in this field with the lowest offset is the Function on the PCIe-MI Component that is furthest upstream from the specified Function and every other byte in this field represents the Function on the path to the specified Function that is on the Secondary Bus of (i.e., downstream from) the Function indicated by the next lower byte offset in this field.↑</p> <p>↑↑If the Path Length field is 0, then this field must not be included in the PCIe-MI Completion .↑</p> |

↑↑The Capability Structure Data Structure is shown in § Table 12-55 .↑

Table 12-55 ↑↑Capability Structure Data Structure↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|------------------|--|
| ↑↑1:0↑ | <p>↑↑Capability ID↑ : ↑↑If the Capability structure is in PCI-compatible Configuration Space (i.e., the first 256 bytes of Configuration Space), then bits 7:0 of this field must indicate the Capability ID and bits 15:8 of this field must be set to FFh.↑</p> <p>↑↑If the Capability structure is in Extended Configuration Space (i.e., beyond the first 256 bytes of Configuration Space), then this field must indicate the Extended Capability ID.↑</p> |
| ↑↑3:2↑ | <p>↑↑Capability Offset↑ : ↑↑This field must indicate the byte offset of the Capability structure relative to offset 0 of the Function's Configuration Space.↑</p> |

12.7.3.2 ↑↑Get Parameters (Opcode 01h) and Set Parameters (Opcode 02h)↑ §

ECN: Base 6.3
PCIe-MI△

↑↑The Get Parameters command returns configurable parameters from a PCIe-MI Component . The Set Parameters command sets configurable parameters on a PCIe-MI Component . The Parameter Identifiers supported by the Get Parameters command and Set Parameters command are listed in § Table 12-56 . Each PCIe-MI Completer in a PCIe-MI Component may support a different set of Parameter Identifiers.↑

Table 12-56 ↑↑Parameter Identifiers↑ §

| ↑↑Parameter Identifier↑ | ↑↑Description↑ | ↑↑Support↑ | ↑↑Section↑ |
|-------------------------|------------------------------------|------------|-------------------------|
| ↑↑00h↑ | ↑↑Configuration Space Writability↑ | ↑↑0↑ | ↑↑§ Section 12.7.3.2.1↑ |

Base 6.4 vs Base 6.3

| ↑↑Parameter Identifier↑ | ↑↑Description↑ | ↑↑Support↑ | ↑↑Section↑ |
|-------------------------|--------------------------|------------|-------------------------|
| ↑↑01h↑ | ↑↑Routing Information↑ | ↑↑O↑ | ↑↑§ Section 12.7.3.2.2↑ |
| ↑↑02h↑ | ↑↑Configuration Space↑ | ↑↑O↑ | ↑↑§ Section 12.7.3.2.3↑ |
| ↑↑03h↑ | ↑↑Connector Subdivision↑ | ↑↑O↑ | ↑↑§ Section 12.7.3.2.4↑ |
| ↑↑04h to BFh↑ | ↑↑Reserved↑ | | |
| ↑↑C0h to FFh↑ | ↑↑Vendor Specific↑ | | |

↑↑Notes↑:

1. ↑↑Primary↓ ↑↑O/M definition: O = optional; M = mandatory over MCTP ↑↑Target – The primary↓ ↑↑for each physical layer binding over which PCIe-MI over MCTP ↑↑target↓ is ↑↑supported and mandatory over MMPT if PCIe-MI over MMPT is supported.↑
2. ↑↑If there are no Configuration Space bits or fields that are writeable using the Set Parameters command with the Configuration Space Parameter Identifier (i.e., Parameter Identifier 02h) to a PCIe-MI Completer, then this Parameter Identifier must not be implemented by that PCIe-MI Completer .↑
3. ↑↑If the PCIe-MI Component does not contain any Downstream Ports, then this Parameter Identifier must not be implemented by any PCIe-MI Completer on the PCIe-MI Component .↑
4. ↑↑If the PCIe-MI Component does not contain any Upstream Ports, then this Parameter Identifier must not be implemented by any PCIe-MI Completer on the PCIe-MI Component .↑

↑↑The PCIe-MI Command format↑ for ↑↓control functions↓ ↑↑the Get Parameters command is shown in § Table 12-57 .↑

Table 12-57 ↑↑Get Parameters Command↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|------------------|--|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field specifies values as per § Table 12-21 with the Opcode field set to 01h (i.e., Get Parameters, see § Table 12-27).↑ |

↑↑Get Parameters Command Content↑

| | |
|------|--|
| ↑↑8↑ | ↑↑Parameter Identifier↑ : ↑↑This field specifies the Parameter Identifier (see § Table 12-56). If this field specifies an unsupported Parameter Identifier, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |
|------|--|

↑↑Get Parameters-Specific Command Content↑

| | |
|----------|--|
| ↑↑N+8:9↑ | ↑↑Get Parameters-Specific Command Content↑ : ↑↑This optional field specifies Parameter Identifier-specific command content as defined in the following subsections, where N is the size↑ of ↑↑this field in bytes. If N is 0, then this field is not present.↑ |
|----------|--|

↑↑The PCIe-MI Command format for↑ the ↑↓adapter↓ ↑↑Set Parameters command is shown in § Table 12-58 .↑

Table 12-58 ↑↑Set Parameters Command↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|------------------|--|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field specifies values as per § Table 12-21 with the Opcode field set to 02h (i.e., Set Parameters, see § Table 12-27).↑ |

↑↑Set Parameters Command Content↑

| ↑↑Byte Location↑ | ↑↑Field Description↑ |
|---|---|
| ↑↑8↑ | ↑↑Parameter Identifier↑ : ↑↑This field specifies the Parameter Identifier (see § Table 12-56). If this field specifies an unsupported Parameter Identifier, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |
| ↑↑Set Parameters-Specific Command Content↑ | ↑↑Set Parameters-Specific Command Content↑ : ↑↑This optional field specifies Parameter Identifier-specific command content as defined in the following subsections, where N is the size of this field in bytes. If N is 0, then this field is not present.↑ |
| ↑↑The PCIe-MI Completion format for the Set Parameters command and Get Parameters command for each Parameter Identifier is defined in the following subsections.↑ | |

12.7.3.2.1 ↑↑Configuration Space Writability (Parameter Identifier 00h)↑

§ ECN: Base 6.3
PCIe-MI△◀▷

↑↑This parameter provides commands that perform the following.↑

- ↑↑Retrieves a list of Configuration Space bits and fields on the specified Function that are writeable using the Set Parameters command with the Configuration Space Parameter Identifier (i.e., Parameter Identifier 02h) on the PCIe-MI Completer on which the Get Parameters command is received (e.g., writeable by a BMC on an out-of-band management transport).↑
 - ↑↑Indicates which of those writeable Configuration Space bits and fields are lockable (i.e., able to be set to read-only) when accessed using a Configuration Write Request (e.g., lockable when accessed by in-band management system software).↑
- ↑↑Sets lockable Configuration Space bits and fields to read-only when accessed using a Configuration Write Request.↑

IMPLEMENTATION NOTE: CONFIGURATION SPACE WRITABILITY EXAMPLE §

↑↑The list of Configuration Space bits and fields that are writeable when accessed using the Set Parameters command with the Configuration Space Parameter Identifier are able to be described by an OSC Control field bit (see the PCI Firmware Specification) or ↑↑major component, primarily↑↑by a Capability structure or Extended Capability structure (see the Configuration Space Writability Data Structure Type field and the Configuration Space Writability Data Structure Information field in § Table 12-61 and § Table 12-66). If all RW, RW1C, RWS, and RW1CS bits and fields in a Capability structure or Extended Capability structure are writeable when ↑↑accessed using the ↑↑component↑↑Set Parameters command with the Configuration Space Parameter Identifier, then this list is able to indicate that Capability structure or ↑↑adapter has main power supplied↑↑Extended Capability structure.↑

↑↑However, there may be cases where the Configuration Space bits and fields that are required to be writeable when accessed using the Set Parameters command with the Configuration Space Parameter Identifier are not able to be described by a Capability structure or Extended Capability structure. For example, using the OSC method, system firmware may retain control of the Completion Timeout Value field and the Completion Timeout Disable bit, while granting control of all other bits and fields in the PCI Express Capability structure↑ (i.e., ↑↑it↓↑↑the PCI Capability structure with a Capability ID of 10h) to the operating system running on the host CPU(s).↑

↑↑A BMC on such a system may require using the Set Parameters command with the Configuration Space Parameter Identifier to write the Completion Timeout Value field and the Completion Timeout Disable bit. A PCIe-MI Completer could support writing all writeable bits and fields in the PCI Express Capability structure using the Set Parameters command with the Configuration Space Parameter Identifier in order to allow the BMC to write to the Completion Timeout Value field and the Completion Timeout Disable bit within the PCI Express Capability structure. However, a PCIe-MI Completer may want to allow the Completion Timeout Value field and the Completion Timeout Disable bit to be writeable using the Set Parameters command with the Configuration Space Parameter Identifier while prohibiting any other bits or fields in the PCI Express Capability structure from being writeable using the Set Parameters command with the Configuration Space Parameter Identifier in order to prevent conflicts with the operating system running on the host CPU(s) that is in control of those other bits and fields.↑

↑↑To satisfy this use case, the PCIe-MI Completer is able to allow a BMC to write the Completion Timeout Value field and the Completion Timeout Disable bit using the Set Parameters command with the Configuration Space Parameter Identifier but prohibit the BMC from writing other bits or fields in the PCI Express Capability structure by advertising support for writability of the bits and fields defined as writeable only by firmware when firmware retains control of the PCI Express Completion Timeout Control bit (i.e., bit 8 of the OSC Control field defined by the PCI Firmware Specification). See the OSC Control Field Bit Offset Value in § Table 12-61 and § Table 12-66 for additional details.↑

12.7.3.2.1.1 ↑↑Get Parameters – Configuration Space Writability↑ §

ECN: Base 6.3
PCIe-MI△◀▷

↑↑This command returns a list of Configuration Space bits and fields on the specified Function that are writeable when accessed using the Set Parameters command with the Configuration Space Parameter Identifier on the PCIe-MI Completer on which the Get Parameters command↑ is ↑↑received (e.g., writeable by a BMC on an out-of-band management transport) and indicates which of the writeable Configuration Space bits and fields are lockable (i.e., able to be set to read-only) when accessed using a Configuration Write Request (e.g., lockable when accessed by in-band management system software).↑

↑↑This command uses the PCIe-MI Command format shown in § Table 12-59.↑

Base 6.4 vs Base 6.3

Table 12-59 **↑↑Get Parameters – Configuration Space Writability Command** §

| ↑↑Byte Location | ↑↑Field Description |
|--|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header : ↑↑This field specifies values per § Table 12-57 .↑ |
| ↑↑Get Parameters - Configuration Space Writability Command Content | |
| ↑↑8↑ | ↑↑Parameter Identifier : ↑↑This field is cleared to 00h to specify the Configuration Space Writability Parameter Identifier (see § Table 12-56).↑ |
| ↑↑Get Parameters - Configuration Space Writability-Specific Command Content | |
| ↑↑12:9↑ | ↑↑Function Identifier : ↑↑This field specifies the Function Identifier. If this field specifies a Function Identifier that does not typically exist, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |

↑↑The PCIe-MI Completion format for this command is shown in § Table 12-60 .↑

Table 12-60 **↑↑Get Parameters – Configuration Space Writability Completion** §

| ↑↑Byte Location | ↑↑Field Description |
|--|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header : ↑↑This field must be set as per § Table 12-24 .↑ |
| ↑↑Get Parameters - Configuration Space Writability Completion Header | |
| ↑↑8↑ | ↑↑Number of Configuration Space Writability Data Structures : ↑↑This field must indicate the number of Configuration Space Writability Data Structures returned (see § Table 12-61).↑ |
| ↑↑9↑ | ↑↑Configuration Space Writability Completion Version : ↑↑This field indicates the version of the Configuration Space Writability Completion Header and Configuration Space Writability Completion Content. This field must be cleared to 0 for this version of the specification.↑ |
| ↑↑10↑ | ↑↑Configuration Space Writability Completion Header Length : ↑↑This field indicates the length in bytes of the Configuration Space Writability Completion Header. This field must be set to 4 for this version of the specification.↑ |
| ↑↑11↑ | ↑↑Configuration Space Writability Data Structure Length : ↑↑This field indicates the length in bytes of each Configuration Space Writability Data Structure (see § Table 12-61). This field must be set to 3 for this version of the specification.↑ |
| ↑↑Get Parameters - Configuration Space Writability Completion Content | |
| ↑↑M+11:12↑ | ↑↑Configuration Space Writability Data Structure 0↑ : ↑↑This field must indicate the first Configuration Space Writability Data Structure (see § Table 12-61), if there is at least one Writeable Configuration Data Structure supported by the specified Function, where M is the value indicated by the Configuration Space Writability Data Structure Length field.↑ |
| ↑↑2*M+11:M+12↑ | ↑↑Configuration Space Writability Data Structure 1↑ : ↑↑This field must indicate the second Configuration Space Writability Data Structure supported by the specified Function, if there is more than one Writeable Configuration Data Structures.↑ |
| ↑↑...↑ | ↑↑...↑ |
| ↑↑(N+1)*M+11:N*M+12↑ | ↑↑Configuration Space Writability Data Structure N↑ : ↑↑This field must indicate the last Configuration Space Writability Data Structure, if there are more than two Writeable Configuration Data Structures supported by the specified Function, where N is the number of Configuration Space Writability Data Structures in the list minus 1.↑ |

↑↑The Configuration Space Writability Data Structure format is shown in § Table 12-61 .↑

Base 6.4 vs Base 6.3

Table 12-61 *Configuration Space Writability Data Structure*

| ↓↑Byte Location↑ | ↓↑Field Description↑ |
|------------------|--|
| | <p>↓↑Configuration Space Writability Data Structure Information↓ : ↓↑This field indicates information about the Configuration Space bits and fields that are writeable↓ when ↓↑accessed using the Set Parameters command with the Configuration Space Parameter Identifier on the PCIe-MI Completer on which the Get Parameters command is received (e.g., writeable by a BMC on an out-of-band management transport) and information about which Configuration Space bits and fields are able to be locked (i.e., configured as read only) when accessed using a Configuration Write Request (e.g., lockable when accessed by in-band management system software).↓</p> |
| ↓↑Bits↑ | ↓↑Description↑ |
| ↓↑7↓ | <p>↓↑Lockable In-band↓ : ↓↑If the Configuration Space Writability Data Structure Type bit is 0b (i.e., a Capability Structure) and all of the writeable Configuration Space bits and fields (i.e., bits and fields that are RW, RW1C, RWS, and RW1CS) in the Capability Structure are able to be locked from modification when accessed using a Configuration Write Request (e.g., by in-band management system software) by setting the Lock bit in a Set Parameters command with the Configuration Space Writability Parameter Identifier (see § Table 12-66), then this bit must be Set; otherwise, this bit must be Clear.↑</p> <p>↓↑If the Configuration Space Writability Data Structure Type bit is 1b (i.e., an _OSC Control field bit) and all of the writeable Configuration Space bits and fields (i.e., bits and fields that are RW, RW1C, RWS, and RW1CS) indicated by the _OSC Control field bit (see the PCI Firmware Specification) as writeable↓ only ↓↑auxiliary power↓ ↓↑by firmware when firmware retains control of the feature (i.e., the operating system↓ is ↓↑supplied). Secondary MCTP Target (The secondary target↓ ↓↑never permitted to write to the bits or fields) are able to be locked from modification when accessed using a Configuration Write Request by setting the Lock bit in the Set Parameters command with the Configuration Space Writability Parameter Identifier, then this bit must be Set; otherwise, this bit must be Clear.↑</p> |
| ↓↑0↓ | <p>↓↑Locked In-band↓ : ↓↑If the lockable bits and fields described by this data structure are locked from modification (i.e., the writeable bits and fields that are able to be locked are configured as read-only) when accessed using a Configuration Write Request (e.g., by in-band management system software), then this bit must be Set; otherwise, this bit must be Clear.↑</p> |
| ↓↑6↓ | <p>↓↑The default value of this bit following a Conventional Reset or Function Level Reset of the specified Function↓ is ↓↑implementation specific (e.g., see the SFI RO In-band Access bit in the SFI Control Register). Any type of reset other than a Conventional Reset or Function Level Reset of the specified Function (e.g., a PCIe-MI Completer Reset caused by the 2-Wire clock being held low↓ for ↓↑control functions↓ ↓↑longer than T_{dcl}) must not cause this bit to change value.↑</p> |
| ↓↑5:1↓ | <p>↓↑Reserved↓</p> |
| | <p>↓↑Configuration Space Writability Data Structure Type↓ : ↓↑This bit must indicate the type of the Configuration Space Writability Data Structure.↓</p> |
| ↓↑Value↓ | ↓↑Description↑ |
| ↓↑0↓ | <p>↓↑Capability Structure↓ : ↓↑The type of the Configuration Space Writability Data Structure is a Capability structure or an Extended Capability structure. All writeable Configuration Space bits and fields (i.e., bits and fields that are RW, RW1C, RWS, and RW1CS) in the indicated Capability structure or Extended Capability structure must be writeable↓ when ↓↑accessed using↓ the ↓↑primary target↓ ↓↑Set Parameters command with the Configuration Space Parameter Identifier on the PCIe-MI Completer on which this Get Parameters command↓ is ↓↑offline (such↓ ↓↑received,↓</p> |
| ↓↑1b↓ | <p>↓↑_OSC Control Field Bit↓ : ↓↑The type of the Configuration Space Writability Data Structure is an _OSC Control field. All writeable Configuration Space bits and fields (i.e.,</p> |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|--|--|--|
| ↑↑Bits↑ | ↑↑Description↑ | |
| ↑↑Value↑ | ↑↑Description↑ | |
| ↑↑Configuration Space Writability Data Structure Offset↑ : ↑↑If the type of the Configuration Space Writability Data Structure is a Capability structure or Extended Capability structure (i.e., the Configuration Space Writability Data Structure Type bit is 0b), then this field must indicate the information in the following table.↑ | | |
| ↑↑Bits↑ | ↑↑Description↑ | |
| ↑↑15:12↑ | ↑↑Reserved↑ | |
| ↑↑11:0↑ | ↑↑Configuration Space Byte Offset↑ : ↑↑This field indicates the starting byte offset of the Capability structure↑ or ↑↑Extended Capability structure.↑ | |
| ↑↑2:1↑ ↑↑If the type of the Configuration Space Writability Data Structure is an _OSC Control field (i.e., the Configuration Space Writability Data Structure Type bit is 1b), then this field must indicate the information in the following table.↑ | | |
| ↑↑Bits↑ | ↑↑Description↑ | |
| ↑↑15:5↑ | ↑↑Reserved↑ | |
| ↑↑4:0↑ | ↑↑OSC Control Field Bit Offset Value↑ : ↑↑This field indicates the value of the _OSC Control field bit offset.↑ ↑↑For example,↑ for ↑↑control functions↑ ↑↑the PCI Express Completion Timeout Control bit, which is defined as bit 8 of the _OSC Control field by the PCI Firmware Specification, this field is set to a value of 8.↑ | |

12.7.3.2.1.2 ↑↑Set Parameters – Configuration Space Writability↑

ECN: Base 6.3
PCIe-MI△

↑↑This command locks the specified Configuration Space bits and fields of the specified Function from being able to be modified (i.e., configures the bits and fields as read only) or unlocks the bits and fields so they are able to be modified when accessed using a Configuration Write Request (e.g., when accessed by in-band management system software). This command uses the PCIe-MI Command format shown in § Table 12-66 .↑

Table 12-66 ↑↑Set Parameters – Configuration Space Writability Command↑

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|--|---|--|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field specifies values per § Table 12-58.↑ | |
| ↑↑Set Parameters - Configuration Space Writability Command Content↑ | | |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑ | ↑↑Field Description↑ | | | | | | |
|--|---|----------|----------------|----------|--|---------|---|
| ↑↑8↑ | ↑↑Parameter Identifier↑ : ↑↑This field is cleared to 00h to specify the Configuration Space Writability Parameter Identifier (see § Table 12-56).↑ | | | | | | |
| ↑↑Set Parameters - Configuration Space Writability-Specific Command Content↑ | | | | | | | |
| ↑↑12:9↑ | ↑↑Function Identifier↑ : ↑↑This field specifies the Function Identifier of the Function on which the Configuration Space bit or field writability when accessed using a Configuration Write Request must be set as specified by the Lock bit in the Configuration Space Writability Data Structure Information field. If this field specifies a Function Identifier that ↑↑does not exist, then an Invalid Input Error Completion with the ↑↑primary target. Examples include controlling Flex I/O functions↑ ↑↑Error Location field indicating this field must be returned.↑ | | | | | | |
| ↑↑Configuration Space Writability Data Structure Information↑ : ↑↑This field specifies the type of the Configuration Space Writability Data Structure and whether the specified Configuration Space bits and fields are to be locked when accessed using a Configuration Write Request.↑ | | | | | | | |
| ↑↑Bits↑ | ↑↑Description↑ | | | | | | |
| ↑↑7↑ | <p>↑↑Lock↑ : ↑↑If this bit is Set and the specified Configuration Space bits and fields contain writeable bits or ↑↑setting fields that are lockable, then the specified writeable Configuration Space bits and fields that are lockable must be locked from modification (i.e., set to read-only) when accessed using a ↑↑desired connector subdivision mode before a Configuration Write Request; otherwise, the ↑↑primary target↑ ↑↑specified writeable Configuration Space bits and fields must not be locked from modification when accessed using a Configuration Write Request.↑</p> <p>↑↑This command specifying to lock Configuration Space bits and fields that are already locked from modification when accessed using a Configuration Write Request or specifying to unlock Configuration Space bits and fields that are already not locked from modification when accessed using a Configuration Write Request↑ is ↑↑powered↑ ↑↑not an error.↑</p> | | | | | | |
| ↑↑6:1↑ | ↑↑Reserved↑ | | | | | | |
| ↑↑13↑ | <p>↑↑Configuration Space Writability Data Structure Type↑ : ↑↑This bit specifies the type of the Configuration Space Writability Data Structure.↑</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #ffffcc;">↑↑Value↑</th><th style="background-color: #ffffcc;">↑↑Description↑</th></tr> </thead> <tbody> <tr> <td style="background-color: #ffffcc;">↑↑0h↑</td><td>↑↑Capability Structure↑ : ↑↑The type of the Configuration Space Writability Data Structure is a Capability structure or ↑↑out an Extended Capability structure.↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↑1h↑</td><td>↑↑ OSC Control Field Bit↑ : ↑↑The type of ↑↑reset. Although the Configuration Space Writability Data Structure is an OSC Control field bit (see the PCI Firmware Specification).↑</td></tr> </tbody> </table> | ↑↑Value↑ | ↑↑Description↑ | ↑↑0h↑ | ↑↑Capability Structure↑ : ↑↑The type of the Configuration Space Writability Data Structure is a Capability structure or ↑↑out an Extended Capability structure.↑ | ↑↑1h↑ | ↑↑ OSC Control Field Bit↑ : ↑↑The type of ↑↑reset. Although the Configuration Space Writability Data Structure is an OSC Control field bit (see the PCI Firmware Specification).↑ |
| ↑↑Value↑ | ↑↑Description↑ | | | | | | |
| ↑↑0h↑ | ↑↑Capability Structure↑ : ↑↑The type of the Configuration Space Writability Data Structure is a Capability structure or ↑↑out an Extended Capability structure.↑ | | | | | | |
| ↑↑1h↑ | ↑↑ OSC Control Field Bit↑ : ↑↑The type of ↑↑reset. Although the Configuration Space Writability Data Structure is an OSC Control field bit (see the PCI Firmware Specification).↑ | | | | | | |
| ↑↑15:14↑ | <p>↑↑Configuration Space Writability Data Structure Offset↑ : ↑↑If the type of the Configuration Space Writability Data Structure is a ↑↑primary target might be typical↑ ↑↑Capability structure or Extended Capability structure (i.e., the Configuration Space Writability Data Structure Type field is 0h), then this field specifies information about the Capability structure or Extended Capability structure as described in the following table.↑</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #ffffcc;">↑↑Bits↑</th><th style="background-color: #ffffcc;">↑↑Description↑</th></tr> </thead> <tbody> <tr> <td style="background-color: #ffffcc;">↑↑15:12↑</td><td>↑↑Reserved↑</td></tr> <tr> <td style="background-color: #ffffcc;">↑↑11:0↑</td><td>↑↑Configuration Space Byte Offset↑ : ↑↑This field specifies the starting byte offset of the Capability structure or Extended Capability structure.↑</td></tr> </tbody> </table> | ↑↑Bits↑ | ↑↑Description↑ | ↑↑15:12↑ | ↑↑Reserved↑ | ↑↑11:0↑ | ↑↑Configuration Space Byte Offset↑ : ↑↑This field specifies the starting byte offset of the Capability structure or Extended Capability structure.↑ |
| ↑↑Bits↑ | ↑↑Description↑ | | | | | | |
| ↑↑15:12↑ | ↑↑Reserved↑ | | | | | | |
| ↑↑11:0↑ | ↑↑Configuration Space Byte Offset↑ : ↑↑This field specifies the starting byte offset of the Capability structure or Extended Capability structure.↑ | | | | | | |

Base 6.4 vs Base 6.3

| ↑↓Byte Location↑ | ↑↓Field Description↑ | |
|--|----------------------|---|
| ↑↓Bits↑ | ↑↓Description↑ | |
| ↑↓If the type of the Configuration Space Writability Data Structure is an _OSC Control field (i.e., the Configuration Space Writability Data Structure Type field is 1h), then this field specifies information about the _OSC Control field as described in the following table.↑ | | |
| ↑↓Bits↑ | ↑↓Description↑ | |
| ↑↓15:5↑ | ↑↓Reserved↑ | |
| ↑↓4:0↑ | | ↑↓_OSC Control Field Bit Offset Value↑ : ↑↓This field specifies the value of the _OSC Control field bit offset.↑ ↑↓For example,↑ for ↑↓most peripheral subsystems,↑ ↑the PCI Express Completion Timeout Control bit, which is defined as bit 8 of the _OSC Control field by the PCI Firmware Specification, this field specifies↑ a ↑↓secondary target might exist if such out-of-band control functions↑ ↑value of 8h.↑ |
| ↑↓If the writeable Configuration Space bits and fields specified by this command↑ are ↑↓necessary.↑ ↑↓not lockable from modification when accessed using a Configuration Write Request, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ | | |

The ↑↓FRU↓ ↑↑PCIe-MI Completion for this command must be set as per § Table 12-24 with no PCIe-MI Message Content field.↑

12.7.3.2.2 ↑↓Routing Information (Parameter Identifier 01h)↑ §

 ECN: Base 6.3
 PCIe-MI△

↑↓This parameter gets the Segment Number and/or downstream bus number range of the specified Downstream Port or causes the specified Downstream Port to generate a downstream Configuration Write Request in order to set the captured Segment Number, Bus Number, and/or Device Number on the specified downstream Function.↑

IMPLEMENTATION NOTE: ROUTING INFORMATION VS. SFI CAM §

↑↓If eSFI is supported and a Function has not captured a Segment Number, Bus Number, and/or Device Number (e.g., due to being hidden from the operating system running on a host CPU with the SFI DPF Control field set to a value of 11b to allow MCTP VDMs), then the SFI CAM may be used to generate a Configuration Write Request to a downstream Function in order to cause the Function to capture a Segment Number, Bus Number, and/or Device Number. Programming the Segment Number, Bus Number, and/or Device Number of the Function enables the BMC to configure the Function using the MCTP over PCIe VDM transport which typically has a higher bandwidth than sideband buses.↑

↑↓If the SFI CAM is not supported, then this parameter provides an alternative mechanism for a BMC to set the captured Segment Number, Bus Number, and/or Device Number of a downstream Function.↑

12.7.3.2.2.1 $\uparrow\downarrow$ Get Parameters – Routing InformationECN: Base 6.3
PCIe-MI $\triangleleft\triangleright$

$\uparrow\downarrow$ This command returns the routing information (i.e., the Segment Number and/or downstream bus number range) of the specified Downstream Port. This command uses the PCIe-MI Command format shown in § Table 12-71.

Table 12-71 $\uparrow\downarrow$ Get Parameters – Routing Information Command

| $\uparrow\downarrow$ Byte Location | $\uparrow\downarrow$ Field Description |
|---|---|
| $\uparrow\downarrow$ 7:0 | $\uparrow\downarrow$ PCIe-MI Message Header : $\uparrow\downarrow$ This field specifies values per § Table 12-57. |
| $\uparrow\downarrow$Get Parameters - Routing Information Command Content | |
| $\uparrow\downarrow$ 8 | $\uparrow\downarrow$ Parameter Identifier : $\uparrow\downarrow$ This field is set to 01h to specify the Routing Information Parameter Identifier (see § Table 12-56). |
| $\uparrow\downarrow$Get Parameters - Routing Information-Specific Command Content | |
| $\uparrow\downarrow$ 12:9 | $\uparrow\downarrow$ Function Identifier : $\uparrow\downarrow$ This field specifies the Function Identifier of the Downstream Port whose routing information is requested. If this field specifies a Function Identifier that is not the Function Identifier of a Downstream Port on the PCIe-MI Component, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned. |

$\uparrow\downarrow$ The PCIe-MI Completion format for this command is shown in § Table 12-72.

Table 12-72 $\uparrow\downarrow$ Get Parameters – Routing Information Completion

| $\uparrow\downarrow$ Byte Location | $\uparrow\downarrow$ Field Description |
|---|---|
| $\uparrow\downarrow$ 7:0 | $\uparrow\downarrow$ PCIe-MI Message Header : $\uparrow\downarrow$ This field must be set as per § Table 12-24. |
| $\uparrow\downarrow$Get Parameters - Routing Information Completion Content | |
| $\uparrow\downarrow$ 8 | $\uparrow\downarrow$ Secondary Bus Number : $\uparrow\downarrow$ This field must indicate the Downstream Port's Secondary Bus Number. |
| $\uparrow\downarrow$ 9 | $\uparrow\downarrow$ Subordinate Bus Number : $\uparrow\downarrow$ This field must indicate the Downstream Port's Subordinate Bus Number. |
| $\uparrow\downarrow$ 11:10 | $\uparrow\downarrow$ Segment : $\uparrow\downarrow$ Number Information: This field indicates information about the Downstream Port's Segment Number. |
| $\uparrow\downarrow$ 15:9 | $\uparrow\downarrow$ Reserved |
| $\uparrow\downarrow$ 8 | $\uparrow\downarrow$ Segment Number Valid : $\uparrow\downarrow$ If the Downstream Port has a Segment Number, then this bit must be Set; otherwise, this bit must be Clear. |
| $\uparrow\downarrow$ 7:0 | $\uparrow\downarrow$ Segment Number : $\uparrow\downarrow$ If the Downstream Port has a Segment Number, then this field must indicate the Downstream Port's Segment Number; otherwise, this field must be cleared to 0. |

12.7.3.2.2.2 $\uparrow\downarrow$ Set Parameters – Routing InformationECN: Base 6.3
PCIe-MI $\triangleleft\triangleright$

$\uparrow\downarrow$ This command must cause the specified Downstream Port to generate a Configuration Write Request to offset 0h of the specified downstream Function's Configuration Space in order to set the captured Segment Number, Bus Number, and/or Device Number of the specified Function. The Configuration Write Request must be Type 0 or Type 1 based on the Configuration Request routing and conversion rules defined in § Section 7.3.3. The size of the

data payload for the Configuration Write Request is permitted to be any value permitted for Configuration Write Requests. The content of the data payload for the Configuration Write Request is permitted to be any value. The Routing ID used in the Configuration Write Request is specified in the Routing ID field. If there is a Segment Number for the Downstream Port and the Link is in Flit Mode, then the Segment Number in the Configuration Write Request must be set to the Downstream Port's Segment Number.

↑↑↑If an error external to the PCIe-MI Component is detected with a Configuration Write Request (e.g., a Configuration Write Request to a Function external to the PCIe-MI Component timed out or was completed with a UR or CA Completion Status), then the PCIe-MI Completer must return a PCIe-MI Completion with the Status field set to 11h (i.e., External Error) and must not signal the error using any other architected mechanisms (e.g., AER).↑

↑↑↑This command uses the PCIe-MI Command format shown in § Table 12-73.↑

Table 12-73 ↑↑Set Parameters - Routing Information Command↑ §

| ↑↑Byte Location↑ | ↑↑Field Description↑ | |
|---|---------------------------|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ | : ↑↑This field specifies values per § Table 12-58.↑ |
| ↑↑Set Parameters - Routing Information Command Content↑ | | |
| ↑↑8↑ | ↑↑Parameter Identifier↑ | : ↑↑This field is set to 01h to specify the Routing Information Parameter Identifier (see § Table 12-56).↑ |
| ↑↑Set Parameters - Routing Information-Specific Command Content↑ | | |
| ↑↑12:9↑ | ↑↑Function Identifier↑ | : ↑↑This field specifies the Function Identifier of the Downstream Port that is requested to generate the Configuration Write Request. If this field specifies a Function Identifier that is not the Function Identifier of a Downstream Port on the PCIe-MI Component, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |
| ↑↑14:13↑ | ↑↑Routing ID↑ | : ↑↑This field specifies the Routing ID that the Downstream Port must use in the Configuration Write Request. If the Bus Number portion of this field (i.e., bits 15:8) specifies a value that is not greater than or equal to the Downstream Port's Secondary Bus Number and less than or equal to the Downstream Port's Subordinate Bus Number, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |

↑↑↑The PCIe-MI Completion for this command must be set as per § Table 12-24 with no PCIe-MI Message Content field.↑

12.7.3.2.3 ↑↑Configuration Space (Parameter Identifier 02h)↑ §

 ECN: Base 6.3
 PCIe-MI △△

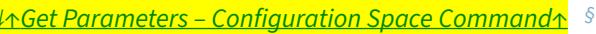
↑↑↑This parameter provides commands that get and set the contents of the specified Configuration Space.↑

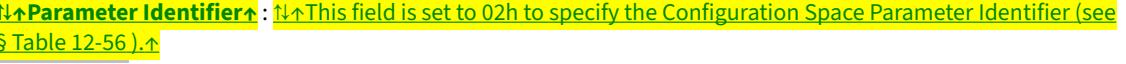
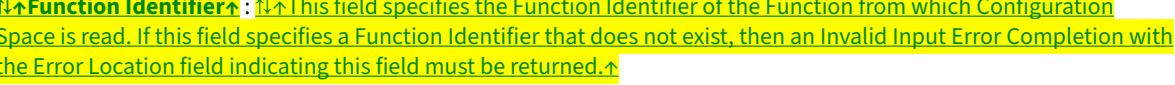
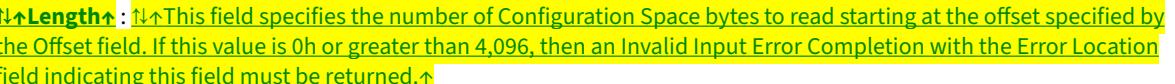
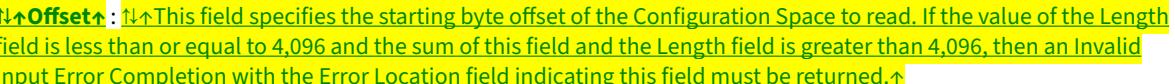
↑↑↑A Function's Configuration Space layout↑ is recommended to ↑↓convey↑ ↑↑be identical when accessed using Configuration Request TLPs and using this parameter (e.g., each Capability structure, Extended Capability structure, register, bit, and field is accessible at the ↑↑presence↓ ↑↑same offset using either mechanism), unless otherwise specified. If a Capability structure, Extended Capability structure, register, bit, or field is accessible using only one of these mechanisms, then that Configuration Space location when accessed from the other mechanism is recommended to be read-only↑ and ↑↓address↓ ↑↑return all 0's when read, unless otherwise specified.↑

↑↑↑If the PCIe-MI Completer supports this parameter on more than one PCIe-MI Completer, then the Configuration Space must be identical when accessed via any↑ of ↑↑those PCIe-MI Completers.↑

ECN: Base 6.3
PCIe-MI△**12.7.3.2.3.1  §**

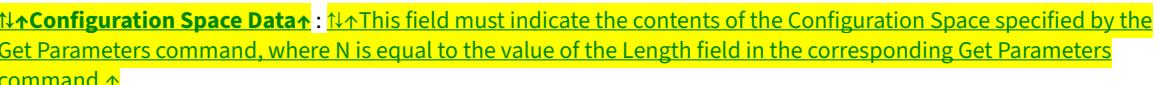
 This command returns the contents of the specified Configuration Space. This command uses the PCIe-MI Command format shown in § Table 12-74. The Configuration Space of a Function that is read using this command may not match the Configuration Space of that Function when read using a Configuration Read Request. For example, if the SFI Extended Capability Hidden bit is Set, then the SFI Extended Capability shows up in the linked list of PCI Express Extended Capabilities when read using this command and does not show up in the linked list of PCI Express Extended Capabilities when read using a Configuration Read Request.

Table 12-74  §

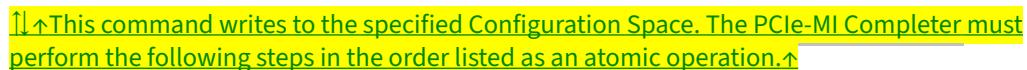
|  |  |
|---|---|
|  |  :  This field specifies values per § Table 12-57. |
| Get Parameters – Configuration Space Command Content | |
|  |  :  This field is set to 02h to specify the Configuration Space Parameter Identifier (see § Table 12-56). |
| Get Parameters – Configuration Space-Specific Command Content | |
|  |  :  This field specifies the Function Identifier of the Function from which Configuration Space is read. If this field specifies a Function Identifier that does not exist, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned. |
|  |  :  This field specifies the number of Configuration Space bytes to read starting at the offset specified by the Offset field. If this value is 0h or greater than 4,096, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned. |
|  |  :  This field specifies the starting byte offset of the Configuration Space to read. If the value of the Length field is less than or equal to 4,096 and the sum of this field and the Length field is greater than 4,096, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned. |

 The PCIe-MI Completion format for this command is shown in § Table 12-75.

Table 12-75  §

|  |  |
|---|---|
|  |  :  This field must be set as per § Table 12-24. |
| Get Parameters – Configuration Space Completion Header | |
|  |  :  This field must indicate the contents of the Configuration Space specified by the Get Parameters command, where N is equal to the value of the Length field in the corresponding Get Parameters command. |

12.7.3.2.3.2  §ECN: Base 6.3
PCIe-MI△

 This command writes to the specified Configuration Space. The PCIe-MI Completer must perform the following steps in the order listed as an atomic operation.

-  Read the DWORD of Configuration Space specified by the Offset field.

2. **↑↑↑Perform a logical AND of the data read in step 1 with the value specified by the AND Mask field.↑**
3. **↑↑↑Perform a logical OR of the data from the result of step 2 with the value specified by the OR Mask field.↑**
4. **↑↑↑Write the DWORD of Configuration Space specified by the Offset field with the data from the result of step 3.↑**

↑↓Forcing control functions **↑↑↑It is not an error if this command specifies a different value than the current value of bits that are not writeable using this command. Those bits are read-only and hence, are not modified by this command.↑**

↑↑↑This command uses the PCIe-MI Command format shown in § Table 12-76 .↑

Table 12-76 **↑↑↑Set Parameters – Configuration Space Command** §

| ↑↑↑Byte Location↑ | ↑↑↑Field Description↑ |
|--|---|
| ↑↑↑7:0↑ | ↑↑↑PCIe-MI Message Header↑ : ↑↑↑This field specifies values per § Table 12-58 .↑ |
| ↑↑↑Set Parameters – Configuration Space Command Content↑ | |
| ↑↑↑8↑ | ↑↑↑Parameter Identifier↑ : ↑↑↑This field is set↑ to revert↑ to 02h↑ to default↑ to specify the Configuration Space Parameter Identifier (see § Table 12-56).↑ |
| ↑↑↑Set Parameters – Configuration Space-Specific Command Content↑ | |
| ↑↑↑12:9↑ | ↑↑↑Function Identifier↑ : ↑↑↑This field specifies the Function Identifier of the Function whose Configuration Space is written. If this field specifies a Function Identifier that does not exist, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |
| ↑↑↑14:13↑ | ↑↑↑Offset↑ : ↑↑↑This field specifies the byte offset of the DWORD of Configuration Space to write. If the value of this field is not DWORD aligned (i.e., bits 1:0 of this field↑ are implementation specific↑ not 00b) or is greater than 0FFCh, then an Invalid Input Error Completion with the Error Location field indicating this field must be returned.↑ |
| ↑↑↑18:15↑ | ↑↑↑AND Mask↑ : ↑↑↑This field specifies the AND mask to apply to the data.↑ |
| ↑↑↑22:19↑ | ↑↑↑OR Mask↑ : ↑↑↑This field specifies the OR mask to apply to the data.↑ |

↑↑↑The PCIe-MI Completion for this command must be set as per § Table 12-24 with no PCIe-MI Message Content field.↑

12.7.3.2.4 **↑↑↑Connector Subdivision (Parameter Identifier 03h)** §

ECN: Base 6.3
PCIe-MI△
↔

↑↑↑This parameter gets or sets the connector subdivisions of an Upstream Port.↑

12.7.3.2.4.1 **↑↑↑Get Parameters – Connector Subdivision** §

ECN: Base 6.3
PCIe-MI△
↔

↑↑↑This command returns the supported upstream-facing connector subdivisions of the PCIe-MI Component . This command uses the PCIe-MI Command format shown↑ in terms↑ in § Table 12-77 .↑

Table 12-77 **↑↑↑Get Parameters – Connector Subdivision Command** §

| ↑↑↑Byte Location↑ | ↑↑↑Field Description↑ |
|---|--|
| ↑↑↑7:0↑ | ↑↑↑PCIe-MI Message Header↑ : ↑↑↑This field specifies values per § Table 12-57 .↑ |
| ↑↑↑Get Parameters - Connector Subdivision Command Content↑ | |

Base 6.4 vs Base 6.3

| | |
|------------------|--|
| ↑↑Byte Location↑ | ↑↑Field Description↑ |
| ↑↑8↑ | ↑↑Parameter Identifier↑ : ↑↑This field is set to 03h to specify the Connector Subdivision Parameter Identifier (see § Table 12-56).↑ |

↑↑The PCIe-MI Completion format for this command is shown in § Table 12-78 .↑

Table 12-78 ↑↑Get Parameters – Connector Subdivision Completion↑ §

| | |
|---|---|
| ↑↑Byte Location↑ | ↑↑Field Description↑ |
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field must be set as per § Table 12-24.↑ |
| ↑↑Get Parameters - Connector Subdivision Completion Content↑ | |
| ↑↑8↑ | ↑↑Current Connector Subdivision Index↑ : ↑↑This field must indicate the index in the Connector Subdivision Combinations Descriptor field of ↑↑when main power↑ ↑↑the connector subdivision that is ↑↑lost, Fundamental Reset, auxiliary power removal or when explicitly changed↑ ↑↑currently set.↑ |
| ↑↑N+8:9↑ | ↑↑Connector Subdivision Combinations Descriptor↑ : ↑↑This field must indicate the supported connector subdivisions in the format specified by ↑↑§ Table 12-20, where N is↑ the ↑↑platform↑ ↑↑size of this field in bytes.↑ |

12.7.3.2.4.2 ↑↑Set Parameters – Connector Subdivision↑ §

ECN: Base 6.3
PCIe-MI△◀

↑↑This command sets the specified connector subdivision. The PCIe-MI Component reverts to the default Connector Subdivision Descriptor only on a Cold Reset as defined by § Section 12.6.3.1 . This command uses the PCIe-MI Command format shown in § Table 12-79 .↑

Table 12-79 ↑↑Set Parameters – Connector Subdivision Command↑ §

| | |
|---|---|
| ↑↑Byte Location↑ | ↑↑Field Description↑ |
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ : ↑↑This field specifies values per § Table 12-58.↑ |
| ↑↑Set Parameters - Connector Subdivision Command Content↑ | |
| ↑↑8↑ | ↑↑Parameter Identifier↑ : ↑↑This field is set to 03h to specify the Connector Subdivision Parameter Identifier (see § Table 12-56).↑ |
| ↑↑Set Parameters - Connector Subdivision-Specific Command Content↑ | |
| ↑↑9↑ | <p>↑↑Connector Subdivision Index↑ : ↑↑This field specifies the index in the Connector Subdivision Combinations Descriptor of the connector subdivision to set↑ (e.g., ↑↑control settings↑ ↑↑a value of 0 specifies Connector Subdivision 0 and a value of 1 specifies Connector Subdivision 1).↑</p> <p>↑↑If this field specifies a valid index other than the index associated with the current link subdivision and all Links in the current and specified connector subdivision↑ are ↑↑permitted↑ ↑↑in DL_Down, then the PCIe-MI Component must set the connector subdivision↑ to ↑↑persist via non-volatile storage).↑ ↑↑the connector subdivision corresponding to the index specified by this field prior to returning the PCIe-MI Completion for this command.↑</p> <p>↑↑If this field specifies a valid index other than the index associated with the current link subdivision and all Links in the current and specified connector subdivision are not in DL_Down, then the following rules apply.↑</p> <ul style="list-style-type: none"> • ↑↑The PCIe-MI Component must return a PCIe-MI Completion for this command with the Status field cleared to 00h (i.e., Success) and then set the connector subdivision to the connector subdivision corresponding to |

Base 6.4 vs Base 6.3

| ↑↑Byte Location↑↑ | ↑↑Field Description↑↑ |
|-------------------|---|
| | <p>the index specified by this field the next time all Links in the current and specified connector subdivision are in DL_Down unless the PCIe-MI Component undergoes a Cold Reset.[↑]</p> <ul style="list-style-type: none"> ↑↑If the connector subdivision is set multiple times while all associated Links are not in DL_Down, then the PCIe-MI Component must set the connector subdivision corresponding to the index specified by this field in the most recent occurrence of this command.[↑] <p>↑↑If this field specifies the index associated with the current link subdivision, then a PCIe-MI Completion for this command with the Status field cleared to 00h (i.e., Success) must be returned.[↑]</p> <p>↑↑If this field specifies an invalid index, an Invalid Input Error Completion with the Error Location field indicating this field must be returned.[↑]</p> |

↑↑The PCIe-MI Completion for this command is shown in § Table 12-80↑

Table 12-80 ↑↑Set Parameters – Connector Subdivision Completion§

| ↑↑Byte Location↑↑ | ↑↑Field Description↑↑ | | | | | | | |
|---|--|---|---------|----------------|--------|-------------|------|---|
| ↑↑7:0↑ | ↑↑PCIe-MI Message Header↑ | : ↑↑This field must be set as per § Table 12-24. [↑] | | | | | | |
| ↑↑Port Information Completion Content↑ | | | | | | | | |
| | ↑↑Status: ↑↑This field indicates additional status for the command. [↑] <table border="1" style="margin-top: 10px;"> <thead> <tr> <th style="background-color: #ffffcc;">↑↑Bits↑</th> <th style="background-color: #ffffcc;">↑↑Description↑</th> </tr> </thead> <tbody> <tr> <td>↑↑7:1↑</td><td>↑↑Reserved↑</td></tr> <tr> <td>↑↑0↑</td><td>↑↑Connector Subdivision Pending↑ : ↑↑If any of the Links associated with the current or specified connector subdivision are not in DL_Down and must be transitioned to DL_Down due to any condition other than a Cold Reset in order to change the connector subdivision, then this bit must be set to 1; otherwise, this bit must be cleared to 0b.[↑]</td></tr> </tbody> </table> | | ↑↑Bits↑ | ↑↑Description↑ | ↑↑7:1↑ | ↑↑Reserved↑ | ↑↑0↑ | ↑↑Connector Subdivision Pending↑ : ↑↑If any of the Links associated with the current or specified connector subdivision are not in DL_Down and must be transitioned to DL_Down due to any condition other than a Cold Reset in order to change the connector subdivision, then this bit must be set to 1; otherwise, this bit must be cleared to 0b. [↑] |
| ↑↑Bits↑ | ↑↑Description↑ | | | | | | | |
| ↑↑7:1↑ | ↑↑Reserved↑ | | | | | | | |
| ↑↑0↑ | ↑↑Connector Subdivision Pending↑ : ↑↑If any of the Links associated with the current or specified connector subdivision are not in DL_Down and must be transitioned to DL_Down due to any condition other than a Cold Reset in order to change the connector subdivision, then this bit must be set to 1; otherwise, this bit must be cleared to 0b. [↑] | | | | | | | |

12.8 Retimer Management §

With increasing interface speeds and varied PCIe topologies, Retimers are expected to be more prevalent. Retimer configuration typically utilizes an SMBus-based dedicated configuration EEPROM with non-standard addressing or required out of band connectivity for in-system updates. Thus, Retimer management is left to the implementer.

12.9 Internal Cable Management §

Internal Cable Management guidance is defined for usage with [CopperLink].

Base 6.4 vs Base 6.3

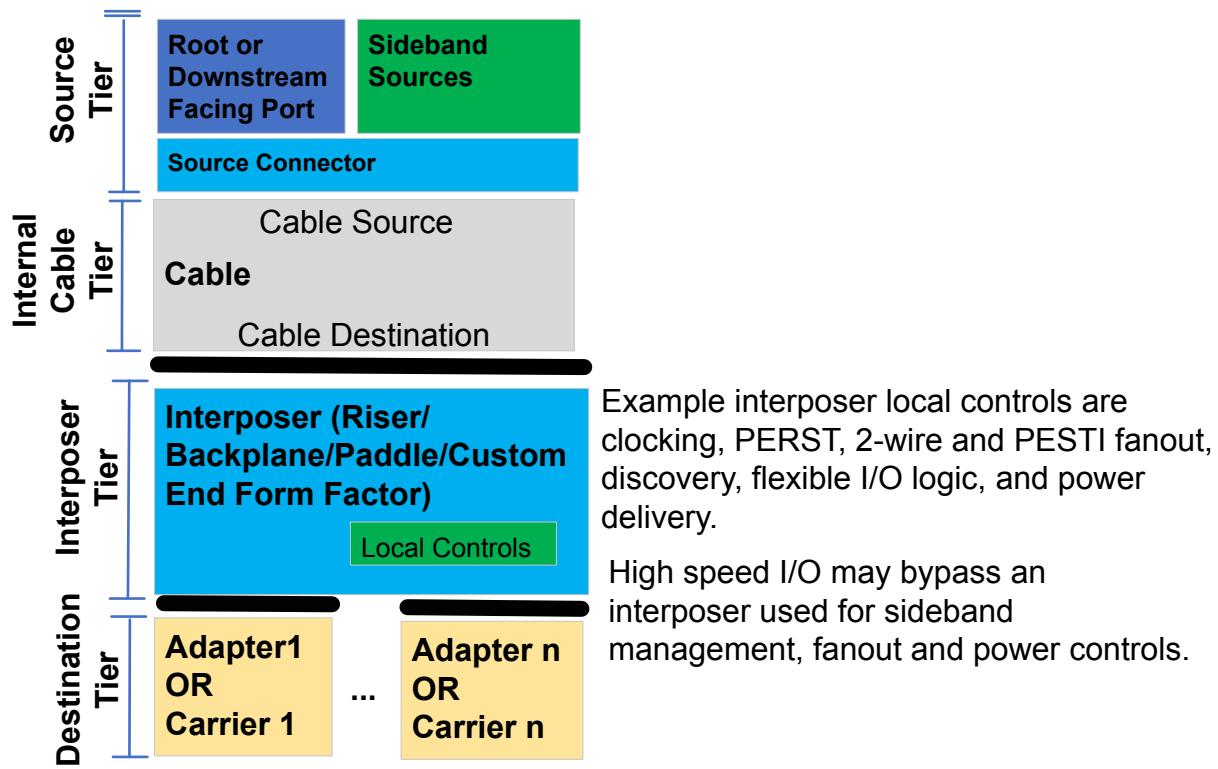


Figure 12-22 Example Tiers Involving Sidebands §

Base 6.4 vs Base 6.3

A. Isochronous Applications §

A.1 Introduction §

The design goal of isochronous mechanisms in PCI Express is to ensure that isochronous traffic receives its allocated bandwidth over a relevant time period while also preventing starvation of other non-isochronous traffic.

Furthermore, there may exist data traffic that requires a level of service falling in between what is required for bulk data traffic and isochronous data traffic. This type of traffic can be supported through the use of Port arbitration within Switches, the use of TC labels [1:7], and optional additional VC resources. Policies for assignment of TC labels and VC resources that are not isochronous-focused are outside the scope of the PCI Express specification.

Two paradigms of PCI Express communication are supported by the PCI Express isochronous mechanisms: Endpoint-to-Root-Complex communication model and peer-to-peer (Endpoint-to-Endpoint) communication model. In the Endpoint-to-Root-Complex communication model, the primary isochronous traffic is memory read and write requests to the Root Complex and read completions from the Root Complex. § Figure A-1 shows an example of a simple system with both communication models. In the figure, devices A, B, called Requesters, are PCI Express Endpoints capable of issuing isochronous request transactions, while device C and Root Complex, called Completers, are capable of being the targets of isochronous request transactions. An Endpoint-to-Root-Complex communication is established between device A and the Root Complex, and a peer-to-peer communication is established between device B and device C. In the rest of this section, Requester and Completer will be used to make reference to PCI Express elements involved in transactions. The specific aspects of each communication model will be called out explicitly.

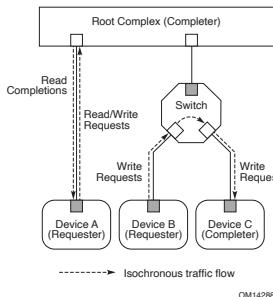


Figure A-1 An Example Showing Endpoint-to-Root-Complex and Peer-to-Peer Communication Models §

Guaranteed bandwidth and deterministic latency require end-to-end configuration of fabric resources. If isochronous traffic is intermixed with non-isochronous traffic, it may not be possible to provide any guarantees/determinism as required by the application usage model. It is recommended that system software configure and assign fabric resources such that traffic intermix either does not occur or is such that the application usage model guarantees can be met. This can be accomplished by assigning dedicated VC resources and corresponding TC labels to the isochronous traffic flow(s) on a given path within the fabric.

Note that there may be one or more isochronous traffic flows per VC/TC label and it is up to system software to insure that the aggregation of these flows does not exceed the requisite bandwidth and latency requirements.

It is also possible for a fabric to support multiple isochronous traffic flows separated across multiple VC (a given flow cannot span multiple VC/TC labels).

In general, as long as the device can meet the isochronous bandwidth and latency requirements, there is nothing to preclude a single VC device from supporting isochronous traffic if multiple TC labels are supported to delineate such traffic from non-isochronous traffic within the fabric.

A.2 Isochronous Contract and Contract Parameters §

In order to support isochronous data transfer with guaranteed bandwidth and deterministic latency, an isochronous contract must be established between a Requester/Completer pair and the PCI Express fabric. This contract must enforce both resource reservation and traffic regulation. Without such a contract, two basic problems, over-subscription and congestion, may occur as illustrated in § Figure A-2 . When interconnect bandwidth resources are over-subscribed, the increased latency may cause failure of isochronous service and starvation of non-isochronous services. Traffic congestion occurs when flow control credits are not returned possibly due to a higher than expected/provisioned packet injection rate. This may cause excessive service latencies for both isochronous traffic and non-isochronous traffic.

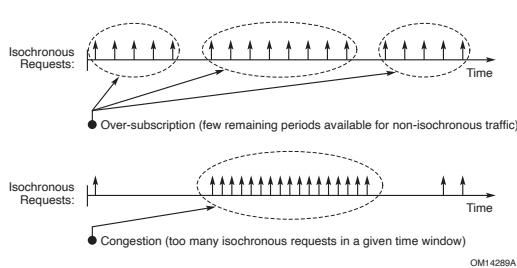


Figure A-2 Two Basic Bandwidth Resourcing Problems: Over-Subscription and Congestion §

The isochronous transfer mechanism in this specification addresses these problems with traffic regulation including admission control and service discipline. Under a software managed admission control, a Requester must not issue isochronous transactions unless the required isochronous bandwidth and resource have been allocated. Specifically, the isochronous bandwidth is given by the following formula:

$$BW = \frac{N * Y}{T}$$

Equation A-1 Isochronous Bandwidth §

The formula defines allocated bandwidth (BW) as a function of specified number (N) of transactions of a specified payload size (Y) within a specified time period (T). Another important parameter in the isochronous contract is latency. Based on the contract, isochronous transactions are completed within a specified latency (L). Once a Requester/Completer pair is admitted for isochronous communication, the bandwidth and latency are guaranteed to the Requester (a PCI Express Endpoint) by the Completer (Root Complex for Endpoint-to-Root-Complex communication and another PCI Express Endpoint for peer-to-peer communication) and by the PCI Express fabric components (Switches).

Specific service disciplines must be implemented by isochronous-capable PCI Express components. The service disciplines are imposed to PCI Express Switches and Completers in such a manner that the service of isochronous requests is subject to a specific service interval (t). This mechanism is used to provide the method of controlling when an isochronous packet injected by a Requester is serviced. Consequently, isochronous traffic is policed in such manner that only packets that can be injected into the fabric in compliance with the isochronous contract are allowed to make immediate progress and start being serviced by the PCI Express fabric. A non-compliant Requester that tries to inject more isochronous transactions than what was being allowed by the contract is prevented from doing so by the

flow-control mechanism thereby allowing compliant Requesters to correctly operate independent of non-compliant Requesters.

In the Endpoint-to-Root-Complex model, since the aggregated isochronous traffic is eventually limited by the host memory subsystem's bandwidth capabilities, isochronous read requests, and write requests (and Messages) are budgeted together. A Requester may divide the isochronous bandwidth between read requests and write requests as appropriate.

A.2.1 Isochronous Time Period and Isochronous Virtual Timeslot §

The PCI Express isochronous time period (T) is uniformly divided into units of virtual timeslots (t). To provide precise isochronous bandwidth distribution only one isochronous request packet is allowed per virtual timeslot. The virtual timeslot supported by a PCI Express component is reported through the Reference Clock field in the Virtual Channel Capability structure or the Multi-Function Virtual Channel Capability structure. When Reference Clock = 00b, duration of a virtual timeslot t is 100 ns. Duration of isochronous time period T depends on the number of phases of the supported time-based WRR Port arbitration table size. When the time-based WRR Port Arbitration Table size equals to 128, there are 128 virtual timeslots (t) in an isochronous time period (i.e., $T = 12.8 \mu\text{s}$).

Note that isochronous period T as well as virtual timeslots t do not need to be aligned and synchronized among different PCI Express isochronous devices, i.e., the notion of $\{T, t\}$ is local to each individual isochronous device.

A.2.2 Isochronous Payload Size §

The payload size (Y) for isochronous transactions must not exceed $\downarrow\downarrow\text{Max_Payload_Size}\downarrow\uparrow\text{Tx_MPS_Limit}$ (see [§ Section 2.2.2](#)). After configuration, the $\downarrow\downarrow\text{Max_Payload_Size}\downarrow\uparrow\text{Tx_MPS_Limit}$ is set and fixed for each path that supports isochronous service with a value required to meet isochronous latency. The fixed $\downarrow\downarrow\text{Max_Payload_Size}\downarrow\uparrow\text{Tx_MPS_Limit}$ 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

$$\uparrow\downarrow Y = \text{Max_Payload_Size}$$

MathType@MTEF@5@5@+-faaagCart1ev2aaaKnaaaaWenf2ys9wBH5garuWqH1MyYLwyaXatLxB19gBaerb9wDYLwzYb
qee0evGueE0jxyaibaiYlf9irVeeu0dXdh9vqqj=hHeeu0xXdbba9frFj0=OqFfea0dXdd9vqaq=JfrVkFHe9pgea0dXdar=Jb9hs0
dXdbPYxe9vr0=vr0=vqpi0dc9GqpWqaaeaabiGaciaacaqabeaadaqaaqaaaOqaaiaadMfacqGH9aqpcqaqGnbGaaeyyiaablha
caqGFbGaaeiuaiaabggacaqG5bGaaeiBaiaab+gacaqGHbGaaeizaab+facaqGtbGaaeyAaiaabQhacaqGLbaaaa@3F23@+

$Y = \text{Tx_MPS_Limit}$

Equation A-2 Isochronous Payload Size §

A transaction with partial writes is treated as a normally accounted transaction. A Completer must account for partial writes as part of bandwidth assignment (for worst case servicing time).

A.2.3 Isochronous Bandwidth Allocation §

Given T , t and Y , the maximum virtual timeslots within a time period is

$$N_{\max} = \frac{T}{t}$$

Equation A-3 N_{\max} §

and the maximum specifiable isochronous bandwidth is

$$BW_{\max} = \frac{Y}{t}$$

Equation A-4 BW_{\max} §

The granularity with which isochronous bandwidth can be allocated is defined as:

$$BW_{\text{granularity}} = \frac{Y}{T}$$

Equation A-5 $BW_{\text{granularity}}$ §

Given T and t at 12.8 μs and 100 ns, respectively, N_{\max} is 128. As shown in § Table A-1, BW_{\max} and $BW_{\text{granularity}}$ 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) | ↑↓1289↓↑↑1280↑ | 2560 | 5120 | 10240 |
| BW _{granularity} (MB/s) | 10 | 20 | 40 | 80 |

Similar to bandwidth budgeting, isochronous service disciplines including arbitration schemes are based on counting requests (not the sizes of those requests). Therefore, assigning isochronous bandwidth BW_{link} to a PCI Express Link is equivalent to assigning N_{link} virtual timeslots per isochronous period, where N_{link} is given by

$$N_{\text{link}} = \frac{BW_{\text{link}}}{BW_{\text{granularity}}}$$

Equation A-6 N_{link} §

A Switch Port serving as an Egress Port (or an RCRB serving as a “virtual” Egress Port) for an isochronous traffic, the N_{\max} virtual timeslots within T are represented by the time-based WRR Port Arbitration Table in the PCI Express Virtual Channel Capability structure detailed in § Section 7.9.1. The table consists of N_{\max} entries. An entry in the table represents one virtual timeslot in the isochronous time period. When a table entry is given a value of PN, it means that the timeslot is assigned to an Ingress Port (in respect to the isochronous traffic targeting the Egress Port) designated by a Port Number of PN. Therefore, N_{link} virtual timeslots are assigned to the Ingress Port when there are N_{link} entries in the table with value of PN. The Egress Port may admit one isochronous request transaction from the Ingress Port for further service only when the table entry reached by the Egress Port’s isochronous time ticker (that increments by 1 every t time

and wraps around when reaching T) is set to PN. Even if there are outstanding isochronous requests ready in the Ingress Port, they will not be served until next round of time-based WRR arbitration. In this manner, the time-based Port Arbitration Table serves for both isochronous bandwidth assignment and isochronous traffic regulation.

For an Endpoint serving as a Requester or a Completer, isochronous bandwidth allocation is accomplished through negotiation between system software and device driver, which is outside of the scope of this specification.

A.2.4 Isochronous Transaction Latency §

Transaction latency is composed of the latency through the PCI Express fabric and the latency contributed by the Completer. Isochronous transaction latency is defined for each transaction and measured in units of virtual timeslot t .

- The *read latency* is defined as the round-trip latency. This is the delay from the time when the device submits a memory read request packet to its Transaction Layer (Transmit side) to the time when the corresponding read completion arrives at the device's Transaction Layer (Receive side).
- The *write latency* is defined as the delay from the time when the Requester posts a memory write request to its PCI Express Transaction Layer (Transmit side) to the time when the data write becomes globally visible within the memory subsystem of the Completer. A write to memory reaches the point of global visibility when all agents accessing that memory address get the updated data.

When the upper bound and the lower bound of isochronous transaction latency are provided, the size of isochronous data buffers in a Requester can be determined. For most of common platforms, the minimum isochronous transaction latency is much smaller than the maximum. As a conservative measure, the minimum isochronous transaction latency is assumed to be zero; only guidelines on measuring the maximum isochronous transaction latency are provided here.

For a Requester, the maximum isochronous (read or write) transaction latency (L) can be accounted as the following:

$$L = L_{Fabric} + L_{Completer}$$

Equation A-7 Max Isochronous Transaction Latency §

where L_{Fabric} is the maximum latency of the PCI Express fabric and $L_{Completer}$ is the maximum latency of the Completer.

L_{Fabric} which applies to both read and write transactions, depends on the topology, latency across each PCI Express Link, and the arbitration point in the path between the Requester to the Completer. The latency on a PCI Express Link depends on pipeline delays, width and operational frequency of the Link, transmission of electrical signals across the medium, wake up latency from low power states, and delays caused by Data Link Layer Retry.

A restriction on the PCI Express topology may be imposed for each targeted platform in order to provide a practically meaningful guideline for L_{Fabric} . The values of L_{Fabric} should be reasonable and serve as practical upper limits under normal operating conditions.

The value of $L_{Completer}$ depends on the memory technology, memory configuration, and the arbitration policies in the Completer that comprehend PCI Express isochronous traffic. The target value for $L_{Completer}$ should provide enough headroom to allow for implementation tradeoffs.

Definitions of read and write transaction latencies for a Completer are different:

- Read transaction latency for the Completer is defined as the delay from the time a memory read transaction is available at the Receiver end of a PCI Express Port in the Completer to the time the corresponding read completion transaction is posted to the transmission end of the PCI Express Port.

- Write transaction latency is defined as the delay from the time a memory write transaction is available at the Receiver end of a PCI Express Port in the Completer to the time that the transmitted data is globally visible.

All of the isochronous transaction latencies defined above are based on the assumption that the Requester injects isochronous transactions uniformly. According to an isochronous contract of $\{N, T, t\}$, the uniform traffic injection is defined such that up to N transactions are evenly distributed over the isochronous period T based on a ticker granularity of virtual timeslot t . For a Requester with non-uniform isochronous transaction injection, the Requester is responsible of accounting for any additional delay due to the deviation of its injection pattern from a uniform injection pattern.

A.2.5 An Example Illustrating Isochronous Parameters §

§ Figure A-3 illustrates the key isochronous parameters using a simplified example with $T = 20t$ and $L = 22t$. A Requester has reserved isochronous bandwidth of four transactions per T . The device shares the allocated isochronous bandwidths for both read requests and write requests. As shown, during one isochronous time period, the Requester issues two read requests and two write requests. All requests are completed within the designated transaction latency L . Also shown in the figure, there is no time dependency between the service time of write requests and the arrival time of read completions.

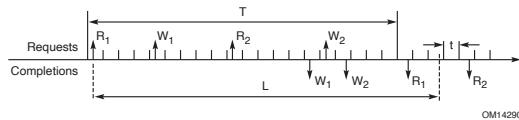


Figure A-3 A Simplified Example Illustrating PCI Express Isochronous Parameters §

A.3 Isochronous Transaction Rules §

Isochronous transactions follow the same rules as described in § Chapter 2. In order to assist the Completer to meet latency requirements, the following additional rules further illustrate and clarify the proper behavior of isochronous transactions:

- The value in the Length field of $\uparrow\downarrow$ read requests must never exceed $\uparrow\downarrow$ Max_Payload_Size. $\downarrow\uparrow$
 $\uparrow\downarrow$ Tx MPS Limit. \uparrow

A.4 Transaction Ordering §

In general, isochronous transactions follow the ordering rules described in § Section 2.4. The following ordering rule further illustrates and clarifies the proper behavior of isochronous transactions:

- There are no ordering guarantees between any isochronous and non-isochronous transactions because the traffic has been segregated into distinct VC resources.
- Isochronous write requests are serviced on any PCI Express Link in strictly the same order as isochronous write requests are posted.
- Switches must allow isochronous posted requests to pass isochronous read completions.

A.5 Isochronous Data Coherency §

Cache coherency for isochronous transactions is an operating system software and Root Complex hardware issue. PCI Express provides the necessary mechanism to control Root Complex behavior in terms of enforcing hardware cache coherency on a per transaction basis.

For platforms where snoop latency in a Root Complex is either unbounded or can be excessively large, in order to meet tight maximum isochronous transaction latency $L_{Completer}$, or more precisely $L_{Root_Complex}$, all isochronous transactions should have the No Snoop Attribute bit set.

A Root Complex must report the Root Complex's capability to the system software by setting the Reject Snoop Transactions field in the VC Resource Capability register (for any VC resource capable of supporting isochronous traffic) in its RCRB. Based on whether or not a Root Complex is capable of providing hardware enforced cache coherency for isochronous traffic while still meeting isochronous latency target, system software can then inform the device driver of Endpoints to set or unset the No Snoop Attribute bit for isochronous transactions.

Note that cache coherency considerations for isochronous traffic do not apply to peer-to-peer communication.

A.6 Flow Control §

Completers and PCI Express fabric components should implement proper sizing of buffers such that under normal operating conditions, no backpressure due to flow control should be applied to isochronous traffic injected uniformly by a Requester. For Requesters that are compliant to the isochronous contract, but have bursty injection behavior, Switches and Completers may apply flow control backpressure as long as the admitted isochronous traffic is uniform and compliant to the isochronous contract. Under abnormal conditions when isochronous traffic jitter becomes significant or when isochronous traffic is oversubscribed due to excessive Data Link Layer Retry, flow control provides a natural mechanism to ensure functional correctness.

A.7 Considerations for Bandwidth Allocation §

A.7.1 Isochronous Bandwidth of PCI Express Links §

Isochronous bandwidth budgeting for PCI Express Links can be derived based on Link parameters such as isochronous payload size and the speed and width of the Link.

Isochronous bandwidth allocation for a PCI Express Link should be limited to certain percentage of the maximum effective Link bandwidth in order to leave sufficient bandwidth for non-isochronous traffic and to account for temporary Link bandwidth reduction due to retries. Link utilization is counted based on the actual cycles consumed on the physical PCI Express Link. The maximum number of virtual slots allowed per Link (N_{link}) depends on the isochronous packet payload size and the speed and width of the Link.

As isochronous bandwidth allocation on a PCI Express Link is based on number of requests N_{link} per isochronous period. There is no distinction between read requests and write requests in budgeting isochronous bandwidth on a PCI Express Link.

A.7.2 Isochronous Bandwidth of Endpoints §

For peer-to-peer communication, the device driver is responsible for reporting to system software if the device is capable of being a Completer for isochronous transactions. In addition, the driver must report the device's isochronous bandwidth capability. The specifics of the report mechanism are outside the scope of this specification.

A.7.3 Isochronous Bandwidth of Switches §

Allocation of isochronous bandwidth for a Switch must consider the capacity and utilization of PCI Express Links associated with the Ingress Port and the Egress Port of the Switch that connect the Requester and the Completer, respectively. The lowest common denominator of the two determines if a requested isochronous bandwidth can be supported.

A.7.4 Isochronous Bandwidth of Root Complex §

Isochronous bandwidth of Root Complex is reported to the software through its RCRB structure. Specifically, the Maximum Time Slots field of the VC Resource Capability register in VC Extended Capability structure indicates the total isochronous bandwidth shared by the Root Ports associated with the RCRB . Details of the platform budgeting for available isochronous bandwidth within a Root Complex are outside of the scope of this specification.

A.8 Considerations for PCI Express Components §

A.8.1 An Endpoint as a Requester §

Before an Endpoint as a Requester can start issuing isochronous request transactions, the following configuration steps must be performed by software:

- Configuration of at least one VC resource capable of supporting isochronous communication and assignment of at least one TC label.
- Enablement of this VC resource.

When the Requester uniformly injects isochronous requests, the Receive Port, either a Switch Port or a Root Port, should issue Flow Control credits back promptly such that no backpressure should be applied to the associated VC. This type of Requester may size its buffer based on the PCI Express fabric latency L_{Fabric} plus the Completer's latency $L_{Completer}$.

When isochronous transactions are injected non-uniformly, either some transactions experience longer PCI Express fabric delay or the Requester gets back-pressed on the associated VC. This type of Requester must size its buffer to account for the deviation of its injection pattern from uniformity.

A.8.2 An Endpoint as a Completer §

An Endpoint may serve as a Completer for isochronous peer-to-peer communication. Before an Endpoint starts serving isochronous transactions, system software must identify/configure a VC resource capable of supporting isochronous traffic and assigned a corresponding TC label.

An Endpoint Completer must observe the maximum isochronous transaction latency ($L_{Completer}$). An Endpoint Completer does not have to regulate isochronous request traffic if attached to a Switch since Switches implement traffic regulation. However, an Endpoint Completer must size its internal buffer such that no backpressure should be applied to the corresponding VC.

A.8.3 Switches §

A Switch may have multiple ports capable of supporting isochronous transactions. Before a Switch starts serving isochronous transactions for a Port, the software must perform the following configuration steps:

- Configuration/enablement of at least one VC resource capable of supporting isochronous communication.
- Configuration of the Port as an Ingress Port:
 - Configuration (or reconfiguration if the associated VC of the Egress Port is already enabled) of the time-based WRR Port Arbitration Table of the targeting Egress Port to include N_{link} entries set to the Ingress Port's Port Number. Here N_{link} is the isochronous allocation for the Ingress Port.
 - Enabling the targeting Egress Port to load newly programmed Port Arbitration Table.
- Configuration of the Port as an Egress Port:
 - Configuration of each VC's Port Arbitration Table with number of entries set according to the assigned isochronous bandwidth for all Ingress Ports.
 - Select proper VC Arbitration, e.g., as strict-priority based VC Arbitration.
 - If required, configuration of the Port's VC Arbitration Table with large weights assigned accordingly to each associated VC.

Each VC associated with isochronous traffic may be served as the highest priority in arbitrating for the shared PCI Express Link resource at an Egress Port. This is comprehended by a Switch's internal arbitration scheme.

In addition, a Switch Port may use “just in time” scheduling mechanism to reduce VC arbitration latency. Instead of pipelining non-isochronous Transport Layer packets to the Data Link Layer of the Egress Port in a manner that Data Link Layer transmit buffer becomes saturated, the Switch Port may hold off scheduling of a new non-isochronous packet to the Data Link Layer as long as it is possible without incurring unnecessary Link idle time.

When a VC configured to support isochronous traffic is enabled for a Switch Port (ingress) that is connected to a Requester, the Switch must enforce proper traffic regulation to ensure that isochronous traffic from the Port conforms to this specification. With such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any non-compliant Requester.

The above isochronous traffic regulation mechanism only applies to request transactions but not to completion transactions. When Endpoint-to-Root-Complex and peer-to-peer communications co-exist in a Switch, an Egress Port may mix isochronous write requests and read completions in the same direction. In the case of contention, the Egress Port must allow write requests to pass read completions to ensure the Switch meets latency requirement for isochronous requests.

A.8.4 Root Complex §

A Root Complex may have multiple Root Ports capable of supporting isochronous transactions. Before a Root Complex starts serving isochronous transactions for a Root Port, the Port must be configured by software to enable VC to support isochronous traffic using the following configuration steps:

- Configuration of at least one VC resource capable of supporting isochronous communication and assignment of at least one TC label.
- Configuration of the Root Port as an Ingress Port:
 - Configuration (or reconfiguration if the associated VC in RCRB is already enabled) of the time-based WRR Port Arbitration Table of the targeting RCRB to include N_{link} entries set to the Ingress Port's Port Number. Here N_{link} is the isochronous allocation for the Port.
 - Enabling the targeting RCRB to load newly programmed Port Arbitration Table.
- Configuration of the Root Port as an Egress Port:
 - If supported, configuration of the Root Port's VC Arbitration Table with large weights assigned to the associated VC.
 - If the Root Complex supports peer-to-peer traffic between Root Ports, configuration of the Root Port's Port Arbitration Table number of entries is set according to the assigned isochronous bandwidth for all Ingress Ports.

A Root Complex must observe the maximum isochronous transaction latency ($L_{Completer}$ or more precisely $L_{Root_Complex}$) that applies to all the Root Ports in the Root Complex. How a Root Complex schedules memory cycles for PCI Express isochronous transactions and other memory transactions is outside of the scope of this specification as long as $L_{Root_Complex}$ is met for PCI Express isochronous transactions.

When a VC is enabled to support isochronous traffic for a Root Port, the Root Complex must enforce proper traffic regulation to ensure that isochronous traffic from the Root Port conforms to this specification. With such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any non-compliant Requesters. Isochronous traffic regulation is implemented using the time-based Port Arbitration Table in RCRB .

Root Complex may perform the following operations for invalid isochronous transactions:

- Return partial completions for read requests with the value in the Length field exceeding
 Max_Payload_Size .
 Tx_MPS_Limit .

B. Symbol Encoding §

§ Table B-1 shows the byte-to-Symbol encodings for data characters. § Table B-2 shows the Symbol encodings for the Special Symbols used for TLP/DLLP Framing and for interface management. RD- and RD+ refer to the Running Disparity of the Symbol sequence on a per-Lane basis.

Table B-1 8b/10b Data Symbol Codes §

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| D0.0 | 00 | 000 00000 | 100111 0100 | 011000 1011 |
| D1.0 | 01 | 000 00001 | 011101 0100 | 100010 1011 |
| D2.0 | 02 | 000 00010 | 101101 0100 | 010010 1011 |
| D3.0 | 03 | 000 00011 | 110001 1011 | 110001 0100 |
| D4.0 | 04 | 000 00100 | 110101 0100 | 001010 1011 |
| D5.0 | 05 | 000 00101 | 101001 1011 | 101001 0100 |
| D6.0 | 06 | 000 00110 | 011001 1011 | 011001 0100 |
| D7.0 | 07 | 000 00111 | 111000 1011 | 000111 0100 |
| D8.0 | 08 | 000 01000 | 111001 0100 | 000110 1011 |
| D9.0 | 09 | 000 01001 | 100101 1011 | 100101 0100 |
| D10.0 | 0A | 000 01010 | 010101 1011 | 010101 0100 |
| D11.0 | 0B | 000 01011 | 110100 1011 | 110100 0100 |
| D12.0 | 0C | 000 01100 | 001101 1011 | 001101 0100 |
| D13.0 | 0D | 000 01101 | 101100 1011 | 101100 0100 |
| D14.0 | 0E | 000 01110 | 011100 1011 | 011100 0100 |
| D15.0 | 0F | 000 01111 | 010111 0100 | 101000 1011 |
| D16.0 | 10 | 000 10000 | 011011 0100 | 100100 1011 |
| D17.0 | 11 | 000 10001 | 100011 1011 | 100011 0100 |
| D18.0 | 12 | 000 10010 | 010011 1011 | 010011 0100 |
| D19.0 | 13 | 000 10011 | 110010 1011 | 110010 0100 |
| D20.0 | 14 | 000 10100 | 001011 1011 | 001011 0100 |
| D21.0 | 15 | 000 10101 | 101010 1011 | 101010 0100 |
| D22.0 | 16 | 000 10110 | 011010 1011 | 011010 0100 |
| D23.0 | 17 | 000 10111 | 111010 0100 | 000101 1011 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |
| 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 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| D20.2 | 54 | 010 10100 | 0010111 0101 | 0010111 0101 |
| D21.2 | 55 | 010 10101 | 1010101 0101 | 1010101 0101 |
| D22.2 | 56 | 010 10110 | 0110101 0101 | 0110101 0101 |
| D23.2 | 57 | 010 10111 | 1110101 0101 | 0001011 0101 |
| D24.2 | 58 | 010 11000 | 1100111 0101 | 0011001 0101 |
| D25.2 | 59 | 010 11001 | 1001101 0101 | 1001101 0101 |
| D26.2 | 5A | 010 11010 | 0101101 0101 | 0101101 0101 |
| D27.2 | 5B | 010 11011 | 1101101 0101 | 0010011 0101 |
| D28.2 | 5C | 010 11100 | 0011101 0101 | 0011101 0101 |
| D29.2 | 5D | 010 11101 | 1011101 0101 | 0100011 0101 |
| D30.2 | 5E | 010 11110 | 0111101 0101 | 1000011 0101 |
| D31.2 | 5F | 010 11111 | 1010111 0101 | 0101001 0101 |
| D0.3 | 60 | 011 00000 | 1001111 0011 | 0110001 1100 |
| D1.3 | 61 | 011 00001 | 0111011 0011 | 1000101 1100 |
| D2.3 | 62 | 011 00010 | 1011011 0011 | 0100101 1100 |
| D3.3 | 63 | 011 00011 | 1100011 1100 | 1100011 0011 |
| D4.3 | 64 | 011 00100 | 1101011 0011 | 0010101 1100 |
| D5.3 | 65 | 011 00101 | 1010011 1100 | 1010011 0011 |
| D6.3 | 66 | 011 00110 | 0110011 1100 | 0110011 0011 |
| D7.3 | 67 | 011 00111 | 1110001 1100 | 0001111 0011 |
| D8.3 | 68 | 011 01000 | 1110011 0011 | 0001101 1100 |
| D9.3 | 69 | 011 01001 | 1001011 1100 | 1001011 0011 |
| D10.3 | 6A | 011 01010 | 0101011 1100 | 0101011 0011 |
| D11.3 | 6B | 011 01011 | 1101001 1100 | 1101001 0011 |
| D12.3 | 6C | 011 01100 | 0011011 1100 | 0011011 0011 |
| D13.3 | 6D | 011 01101 | 1011001 1100 | 1011001 0011 |
| D14.3 | 6E | 011 01110 | 0111001 1100 | 0111001 0011 |
| D15.3 | 6F | 011 01111 | 0101111 0011 | 1010001 1100 |
| D16.3 | 70 | 011 10000 | 0110111 0011 | 1001001 1100 |
| D17.3 | 71 | 011 10001 | 1000111 1100 | 1000111 0011 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |
| 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 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |
| 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 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |
| 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 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |
| 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 |

Base 6.4 vs Base 6.3

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |

Table B-2 8b/10b Special Character Symbol Codes §

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| 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 |

| Data Byte Name | Data Byte Value (hex) | Bits HGF EDCBA (binary) | Current RD- abcdei fghj (binary) | Current RD+ abcdei fghj (binary) |
|----------------|-----------------------|-------------------------|----------------------------------|----------------------------------|
| K28.5 | BC | 101 11100 | 001111 1010 | 110000 0101 |
| K28.6 | DC | 110 11100 | 001111 0110 | 110000 1001 |
| K28.7 | FC | 111 11100 | 001111 1000 | 110000 0111 |
| K23.7 | F7 | 111 10111 | 111010 1000 | 000101 0111 |
| K27.7 | FB | 111 11011 | 110110 1000 | 001001 0111 |
| K29.7 | FD | 111 11101 | 101110 1000 | 010001 0111 |
| K30.7 | FE | 111 11110 | 011110 1000 | 100001 0111 |

C. Physical Layer Appendix §

C.1 8b/10b Data Scrambling Example §

The following subroutines encode and decode an 8-bit value contained in “inbyte” with the LFSR. This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the “C” programming language conventions, where “<<” and “>>” represent the shift left and shift right operators, “>” is the compare greater than operator, and “^” is the exclusive or operator, and “&” is the logical “AND” operator.

```
/*
  this routine implements the serial descrambling algorithm in parallel form
  for the LSFR polynomial: x^16+x^5+x^4+x^3+1
  this advances the LSFR 8 bits every time it is called
  this requires fewer than 25 xor gates to implement (with a static register)
  The XOR required to advance 8 bits/clock is:
bit 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
  8   9  10 11 12 13 14 15  0   1   2   3   4   5   6   7
    8   9  10 11 12 13 14 15
      8   9  10 11 12 13 14 15
        8   9  10 11 12 13 14 15
          The serial data is just the reverse of the upper byte:
bit 0  1  2  3  4  5  6  7
  15 14 13 12 11 10  9   8
*/

```

Base 6.4 vs Base 6.3

```

int scramble_byte(int inbyte)
{
    static int scrambit[16];
    static int bit[16];
    static int bit_out[16];
    static unsigned short lfsr = 0xffff; // 16 bit short for polynomial
    int i, outbyte;

    if (inbyte == COMMA)           // if this is a comma
    {
        lfsr = 0xffff;            // reset the LFSR
        return (COMMA);          // and return the same data
    }

    if (inbyte == SKIP)           // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16; i++)         // convert LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

        for (i=0; i<8; i++)      // convert byte to be scrambled for legibility
        scrambit[i] = (inbyte >> i) & 1;

        // apply the xor to the data
    if (! (inbyte & 0x100) &&      // if not a KCODE, scramble the data
        ! (TrainingSequence == TRUE)) // and if not in the middle of
    {                                // a training sequence
        scrambit[0] ^= bit[15];
        scrambit[1] ^= bit[14];
        scrambit[2] ^= bit[13];
        scrambit[3] ^= bit[12];
        scrambit[4] ^= bit[11];
        scrambit[5] ^= bit[10];
        scrambit[6] ^= bit[9];
        scrambit[7] ^= bit[8];
    }

    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3] ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4] ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];

    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);
}

```

```
outbyte = 0;
for (i= 0; i<8; i++)      // convert data back to an integer
    outbyte += (scrambit[i] << i);

return outbyte;
}
```

Base 6.4 vs Base 6.3

Base 6.4 vs Base 6.3

```

/*
 * NOTE THAT THE DESCRAMBLE ROUTINE IS IDENTICAL TO THE SCRAMBLE ROUTINE
 * this routine implements the serial descrambling algorithm in parallel form
 * this advances the lfsr 8 bits every time it is called
 * this uses fewer than 25 xor gates to implement (with a static register)
 * The XOR tree is the same as the scrambling routine
 */

int unscramble_byte(int inbyte)
{
    static int descrambit[8];
    static int bit[16];
    static int bit_out[16];
    static unsigned short lfsr = 0xffff; // 16 bit short for polynomial
    int outbyte, i;

    if (inbyte == COMMA) // if this is a comma
    {
        lfsr = 0xffff; // reset the LFSR
        return (COMMA); // and return the same data
    }
    if (inbyte == SKIP) // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16; i++) // convert the LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++) // convert byte to be de-scrambled for legibility
        descrambit[i] = (inbyte >> i) & 1;

    // apply the xor to the data
    if (!(inbyte & 0x100) && // if not a KCODE, scramble the data
        !(TrainingSequence == TRUE)) // and if not in the middle of
    { // a training sequence
        descrambit[0] ^= bit[15];
        descrambit[1] ^= bit[14];
        descrambit[2] ^= bit[13];
        descrambit[3] ^= bit[12];
        descrambit[4] ^= bit[11];
        descrambit[5] ^= bit[10];
        descrambit[6] ^= bit[9];
        descrambit[7] ^= bit[8];
    }

    // Now advance the LFSR 8 serial clocks
    bit_out[0] = bit[8];
    bit_out[1] = bit[9];
    bit_out[2] = bit[10];
    bit_out[3] = bit[11] ^ bit[8];
    bit_out[4] = bit[12] ^ bit[9] ^ bit[8];
    bit_out[5] = bit[13] ^ bit[10] ^ bit[9] ^ bit[8];
    bit_out[6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[9];
    bit_out[7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[8] = bit[0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[9] = bit[1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[3] ^ bit[15] ^ bit[14];
    bit_out[12] = bit[4] ^ bit[15];
    bit_out[13] = bit[5];
    bit_out[14] = bit[6];
    bit_out[15] = bit[7];

    lfsr = 0;
    for (i=0; i < 16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i=0; i < 8; i++) // convert data back to an integer
        outbyte += (descrambit[i] << i);

    return outbyte;
}

```

The initial 16-bit values of the LFSR for the first 128 LFSR advances following a reset are listed below:

Base 6.4 vs Base 6.3

| | 0, 8 | 1, 9 | 2, A | 3, B | 4, C | 5, D | 6, E | 7, F |
|----|------|------|------|------|------|------|------|------|
| 00 | FFFF | E817 | 0328 | 284B | 4DE8 | E755 | 404F | 4140 |
| 08 | 4E79 | 761E | 1466 | 6574 | 7DBD | B6E5 | FDA6 | B165 |
| 10 | 7D09 | 02E5 | E572 | 673D | 34CF | CB54 | 4743 | 4DEF |
| 18 | E055 | 40E0 | EE40 | 54BE | B334 | 2C7B | 7D0C | 07E5 |
| 20 | E5AF | BA3D | 248A | 8DC4 | D995 | 85A1 | BD5D | 4425 |
| 28 | 2BA4 | A2A3 | B8D2 | CBF8 | EB43 | 5763 | 6E7F | 773E |
| 30 | 345F | 5B54 | 5853 | 5F18 | 14B7 | B474 | 6CD4 | DC4C |
| 38 | 5C7C | 70FC | F6F0 | E6E6 | F376 | 603B | 3260 | 64C2 |
| 40 | CB84 | 9743 | 5CBF | B3FC | E47B | 6E04 | 0C3E | 3F2C |
| 48 | 29D7 | D1D1 | C069 | 7BC0 | CB73 | 6043 | 4A60 | 6FFA |
| 50 | F207 | 1102 | 01A9 | A939 | 2351 | 566B | 6646 | 4FF6 |
| 58 | F927 | 3081 | 85B0 | AC5D | 478C | 82EF | F3F2 | E43B |
| 60 | 2E04 | 027E | 7E72 | 79AE | A501 | 1A7D | 7F2A | 2197 |
| 68 | 9019 | 0610 | 1096 | 9590 | 8FCD | D0E7 | F650 | 46E6 |
| 70 | E8D6 | C228 | 3AB2 | B70A | 129F | 9CE2 | FC3C | 2B5C |
| 78 | 5AA3 | AF6A | 70C7 | CDF0 | E3D5 | C0AB | B9C0 | D9C1 |

An 8-bit value of 0 repeatedly encoded with the LFSR after reset produces the following consecutive 8-bit values:

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | FF | 17 | C0 | 14 | B2 | E7 | 02 | 82 | 72 | 6E | 28 | A6 | BE | 6D | BF | 8D |
| 10 | BE | 40 | A7 | E6 | 2C | D3 | E2 | B2 | 07 | 02 | 77 | 2A | CD | 34 | BE | E0 |
| 20 | A7 | 5D | 24 | B1 | 9B | A1 | BD | 22 | D4 | 45 | 1D | D3 | D7 | EA | 76 | EE |
| 30 | 2C | DA | 1A | FA | 28 | 2D | 36 | 3B | 3A | 0E | 6F | 67 | CF | 06 | 4C | 26 |
| 40 | D3 | E9 | 3A | CD | 27 | 76 | 30 | FC | 94 | 8B | 03 | DE | D3 | 06 | 52 | F6 |
| 50 | 4F | 88 | 80 | 95 | C4 | 6A | 66 | F2 | 9F | 0C | A1 | 35 | E2 | 41 | CF | 27 |
| 60 | 74 | 40 | 7E | 9E | A5 | 58 | FE | 84 | 09 | 60 | 08 | A9 | F1 | 0B | 6F | 62 |
| 70 | 17 | 43 | 5C | ED | 48 | 39 | 3F | D4 | 5A | F5 | 0E | B3 | C7 | 03 | 9D | 9B |
| 80 | 8B | 0D | 8E | 5C | 33 | 98 | 77 | AE | 2D | AC | 0B | 3E | DA | 0B | 42 | 7A |
| 90 | 7C | D1 | CF | A8 | 1C | 12 | EE | 41 | C2 | 3F | 38 | 7A | 0D | 69 | F4 | 01 |
| A0 | DA | 31 | 72 | C5 | A0 | D7 | 93 | 0E | DC | AF | A4 | 55 | E7 | F0 | 72 | 16 |
| B0 | 68 | D5 | 38 | 84 | DD | 00 | CD | 18 | 9E | CA | 30 | 59 | 4C | 75 | 1B | 77 |

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C0 | 31 | C5 | ED | CF | 91 | 64 | 6E | 3D | FE | E8 | 29 | 04 | CF | 6C | FC | C4 |
| D0 | 0B | 5E | DA | 62 | BA | 5B | AB | DF | 59 | B7 | 7D | 37 | 5E | E3 | 1A | C6 |
| E0 | 88 | 14 | F5 | 4F | 8B | C8 | 56 | CB | D3 | 10 | 42 | 63 | 04 | 8A | B4 | F7 |
| F0 | 84 | 01 | A0 | 01 | 83 | 49 | 67 | EE | 3E | 2A | 8B | A4 | 76 | AF | 14 | D5 |
| 100 | 4F | AC | 60 | B6 | 79 | D6 | 62 | B7 | 43 | E7 | E5 | 2A | 40 | 2C | 6E | 7A |
| 110 | 56 | 61 | 63 | 20 | 6A | 97 | 4A | 38 | 05 | E5 | DD | 68 | 0D | 78 | 4C | 53 |
| 120 | 8B | D6 | 86 | 57 | B2 | AA | 1A | 80 | 18 | DC | BA | FC | 03 | A3 | 4B | 30 |

At 2.5 GT/s, scrambling produces the power spectrum (in the 10-bit domain) shown in § Figure C-1 .

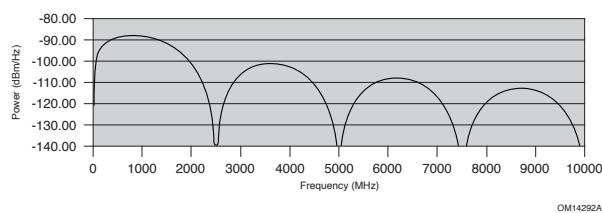


Figure C-1 Scrambling Spectrum at 2.5 GT/s for Data Value of 0 §

C.2 128b/130b Data Scrambling Example §

The following subroutines illustrate how calculate the next value of the 128b/130b scrambling LFSR and how to scramble (or descramble) an 8-bit value, both given the current value of the LFSR.

This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the “C” programming language conventions, where “<<” and “>>” represent the shift left and shift right operators, “^” is the exclusive or operator, and “|=“ is the assignment by bitwise or operator.

Base 6.4 vs Base 6.3

```

#include <stdio.h>

// "Set Bit" - Sets bit number "bit" of "var" to the value "val". Bit "bit" of "var" must start cleared.
#define SB(var,bit,val) var |= (val & 1) << bit

// "Get Bit" - Returns the value of bit number "bit" of "var".
#define GB(var,bit) ((var >> bit) & 1)

// Function to advance the LFSR value by 8 bits, given the current LFSR value
unsigned long int calc_next_lfsr(unsigned long int lfsr) {
    unsigned long int next_lfsr = 0;
    SB(next_lfsr,22, GB(lfsr,14) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(next_lfsr,21, GB(lfsr,13) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21));
    SB(next_lfsr,20, GB(lfsr,12) ^ GB(lfsr,19) ^ GB(lfsr,21));
    SB(next_lfsr,19, GB(lfsr,11) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(next_lfsr,18, GB(lfsr,10) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21));
    SB(next_lfsr,17, GB(lfsr, 9) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(next_lfsr,16, GB(lfsr, 8) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(next_lfsr,15, GB(lfsr, 7) ^ GB(lfsr,22));
    SB(next_lfsr,14, GB(lfsr, 6) ^ GB(lfsr,21));
    SB(next_lfsr,13, GB(lfsr, 5) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(next_lfsr,12, GB(lfsr, 4) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(next_lfsr,11, GB(lfsr, 3) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(next_lfsr,10, GB(lfsr, 2) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(next_lfsr, 9, GB(lfsr, 1) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21));
    SB(next_lfsr, 8, GB(lfsr, 0) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,20));
    SB(next_lfsr, 7, GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21));
    SB(next_lfsr, 6, GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(next_lfsr, 5, GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(next_lfsr, 4, GB(lfsr,17));
    SB(next_lfsr, 3, GB(lfsr,16));
    SB(next_lfsr, 2, GB(lfsr,15) ^ GB(lfsr,22));
    SB(next_lfsr, 1, GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(next_lfsr, 0, GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
    return(next_lfsr);
}

// Function to scramble a byte, given the current LFSR value
unsigned char scramble_data(unsigned long lfsr, unsigned char data_in) {
    unsigned char data_out = 0;
    SB(data_out, 7, GB(data_in,7) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
    SB(data_out, 6, GB(data_in,6) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(data_out, 5, GB(data_in,5) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21));
    SB(data_out, 4, GB(data_in,4) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(data_out, 3, GB(data_in,3) ^ GB(lfsr,19) ^ GB(lfsr,21));
    SB(data_out, 2, GB(data_in,2) ^ GB(lfsr,20) ^ GB(lfsr,22));
    SB(data_out, 1, GB(data_in,1) ^ GB(lfsr,21));
    SB(data_out, 0, GB(data_in,0) ^ GB(lfsr,22));
    return(data_out);
}

// Example of LFSR and scrambled data "0" sequence for Lane 0
main() {
    unsigned long lfsr = 0x1DBFBC; // Lane 0 reset LFSR value
    unsigned char unscrambled_data = 0x00;
    int i;
    printf("Iteration LFSR Next LFSR Scrambled Data\n");
    printf("-----\n");
    for (i = 0; i < 128; i++) {
        unsigned char scrambled_data = scramble_data(lfsr,unscrambled_data);
        unsigned long next_lfsr = calc_next_lfsr(lfsr);
        printf("%3d %06X %06X %02X\n", i, lfsr, next_lfsr, scrambled_data);
        lfsr = next_lfsr;
    }
}

```

The initial 23-bit values of the LFSR for the first 128 LFSR advances for an 8-bit quantity following a reset for Lane 0 are listed below (ordered left to right, top to bottom):

| | 0, 8 | 1, 9 | 2, A | 3, B | 4, C | 5, D | 6, E | 7, F |
|----|--------|--------|--------|--------|--------|--------|--------|--------|
| 00 | 1DBFBC | 498C2E | 1186E9 | 0FC5AD | 7CB75D | 3D8DA2 | 0ECC8F | 379717 |
| 08 | 046BDF | 21D462 | 423FCE | 5177D6 | 1447EF | 67CBA0 | 794F7A | 6CA2AF |

Base 6.4 vs Base 6.3

| | 0, 8 | 1, 9 | 2, A | 3, B | 4, C | 5, D | 6, E | 7, F |
|----|--------|--------|--------|--------|--------|--------|--------|--------|
| 10 | 425060 | 3ED9D6 | 3DBF74 | 3C1A8F | 7AE2E0 | 073E71 | 137D99 | 70B139 |
| 18 | 44F521 | 559720 | 1FBBA8 | 689D9F | 376B02 | 597FFA | 297C0E | 7E450A |
| 20 | 4BDE36 | 669B58 | 6BB530 | 78C3F9 | 0322C0 | 45C7FB | 254F6A | 323D89 |
| 28 | 07FDD2 | 71DFBC | 68726B | 799E27 | 1CFE8A | 4AB864 | 42CB12 | 04AAF3 |
| 30 | 41F947 | 51F808 | 3A98CA | 36A816 | 796895 | 4B4DAF | 54037D | 68E5C7 |
| 38 | 4F3302 | 60A51F | 3AFCE3 | 528116 | 248131 | 1F65E6 | 17D2BA | 34987E |
| 40 | 6C0524 | 44DA45 | 7AF320 | 16FE71 | 5A5134 | 60B5F5 | 2A16E3 | 73AFF1 |
| 48 | 3D3ADA | 18B5AA | 4B9306 | 2BAB58 | 2D179E | 5FDE68 | 46E1F8 | 644B91 |
| 50 | 3F78A4 | 7FCE1B | 23CC59 | 7F017F | 4DA97C | 7EDF8B | 705E13 | 0ADE04 |
| 58 | 6F1775 | 318CBE | 70CC0C | 39C021 | 0944ED | 33F8AB | 21DCBD | 4AE0CE |
| 60 | 1A6112 | 1B2F92 | 17ADD8 | 4BFA7E | 42D358 | 1CE0F3 | 54C164 | 0BFDE2 |
| 68 | 0EF33F | 082717 | 1200E1 | 4FCB73 | 39D53A | 1C5FED | 4ADE41 | 24EE12 |
| 70 | 7046E6 | 122B04 | 642E73 | 5A9AA4 | 0A24D0 | 34C250 | 362B24 | 5B5BB0 |
| 78 | 2833BF | 73F640 | 648BDA | 5E3281 | 490B97 | 373ECC | 0CB1FA | 6FE7A6 |

An 8-bit value of 0x00 repeatedly encoded with the LFSR after reset for Lane 0 produces the following consecutive 8-bit values (ordered left to right, top to bottom):

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 6C | BD | 94 | 98 | 53 | C6 | D8 | CE | 50 | 6A | 75 | C1 | 04 | 4F | C3 | 07 |
| 10 | 75 | 26 | C6 | 06 | A3 | B0 | B4 | AB | 05 | 11 | CC | 57 | 4E | 69 | 42 | 73 |
| 20 | 1D | 0F | B7 | 03 | E0 | 45 | BA | 5E | 30 | EB | D7 | 43 | 2C | 5D | F5 | D0 |
| 30 | 15 | 41 | 76 | 8E | C3 | 9D | D1 | 57 | CD | FF | 76 | A1 | 7A | 4C | 64 | 2E |
| 40 | 87 | 05 | A3 | 24 | 89 | FF | A2 | 4B | 46 | 7C | 1D | 62 | 12 | 19 | A5 | 2F |
| 50 | E6 | B3 | CA | 33 | ED | F3 | 2B | 88 | 67 | 3E | AB | 96 | E8 | 9E | 6A | 5D |
| 60 | 5C | 1C | 64 | 1D | F5 | 2C | 51 | C8 | D8 | A8 | F4 | 4D | 96 | AC | 5D | 7A |
| 70 | 2B | F4 | 2F | 09 | 08 | 2E | 0E | C9 | 02 | 4B | AF | D9 | 3D | 4E | 78 | E7 |

D. Request Dependencies §

This specification does not specify the rules governing the creation of resource dependencies between TLPs using different Traffic Classes. Dependencies between packets in different Traffic Classes can create potential deadlock if devices make different assumptions about what is allowed and what is not. Dependencies can be created when a packet is forwarded (transmitted verbatim) or translated (transmitted with modification) from an input Port to an output Port.

Resource dependencies are created when received packets are forwarded/translated on the same or a different Link. Due to the fact that the forwarding/translating device has finite buffer resources this behavior creates a dependency between the ability to receive a packet and the ability to transmit a packet (in potentially a different VC or sub-channel).

The following notation is used to create a framework to enumerate the possibilities:

$X(m) \rightarrow Y(n)$

This means:

- a request in sub-channel $X(TC = m)$ is forwarded/translated in sub-channel $Y(TC = n)$.
- n and m are between 0-7.
- X and Y are either **P** (Posted Request), **N** (Non-Posted Request), or **C** (Completion).

The list of possible dependencies is:

$P(m) \rightarrow P(n)$

$P(m) \rightarrow N(n)$

$P(m) \rightarrow C(n)$

$N(m) \rightarrow P(n)$

$N(m) \rightarrow N(n)$

$N(m) \rightarrow C(n)$

$C(m) \rightarrow P(n)$

$C(m) \rightarrow N(n)$

$C(m) \rightarrow C(n)$

For a given system, each of these dependencies needs to be classified as legal or illegal for each of the following cases:

- Root Port forwarding to own Link.
- Root Port forwarding to different Root Port's Link.
- Endpoint or Bridge forwarding to own Link.

A Switch is not allowed to modify the TLPs that flow through it, but must ensure complete independence of resources assigned to separate VCs. System software must comprehend the system dependency rules when configuring TC/VC mapping throughout the system.

One possible legal mapping is:

| | RC (Same Port) | RC (Different Port) | Endpoint |
|-------------------------|-------------------|------------------------|----------|
| $P(m) \rightarrow P(n)$ | $m \leq n$ | $m \leq n$ | $m < n$ |
| $P(m) \rightarrow N(n)$ | $m < n$ | $m < n$ | $m < n$ |
| $P(m) \rightarrow C(n)$ | illegal | illegal | illegal |
| $N(m) \rightarrow P(n)$ | $m < n$ | $m < n$ | $m < n$ |

| | RC (Same Port) | RC (Different Port) | Endpoint |
|-------------------------|-------------------|------------------------|----------|
| $N(m) \rightarrow N(n)$ | $m \leq n$ | $m \leq n$ | $m < n$ |
| $N(m) \rightarrow C(n)$ | $m = n$ | $m = n$ | $m = n$ |
| $C(m) \rightarrow P(n)$ | illegal | illegal | illegal |
| $C(m) \rightarrow N(n)$ | illegal | illegal | illegal |
| $C(m) \rightarrow C(n)$ | $m \geq n$ | $m \geq n$ | $m > n$ |

Note that this discussion only deals with avoiding the deadlock caused by the creation of resource dependencies. It does not deal with the additional livelock issues (or lack of forward progress) caused by the system's Virtual Channel arbitration policies.

Some of these potential dependencies are illegal or unreachable:

- $P(m) \rightarrow P(n), N(m) \rightarrow N(n)$
 - $m = n$ - This case is illegal and will lead to deadlock, except when a Request is being forwarded from one Port to another of a Switch or Root Complex.
 - $m \neq n$ - See discussion below.
- $P(m) \rightarrow N(n)$
 - $m = n$ - This case is illegal and will lead to deadlock.
 - $m \neq n$ - See discussion below.
- $N(m) \rightarrow P(n)$ - See discussion below.
- $P(m) \rightarrow C(n)$ - This case is illegal and will lead to deadlock.
- $N(m) \rightarrow C(n)$
 - $m = n$ - This case occurs during the normal servicing of a non-posted request by either a root complex or an endpoint.
 - $m \neq n$ - This case is unreachable and should never happen. Completions always use the same TC as the corresponding request.
- $C(m) \rightarrow P(n), C(m) \rightarrow N(n)$ - These cases are unreachable and should never happen due to the fact that completion buffers must be preallocated.
- $C(m) \rightarrow C(n)$
 - $m = n$ - This case is illegal and will lead to deadlock, except when a Completion is being forwarded from one Port to another of a Switch or Root Complex.
 - $m \neq n$ - This case will occur if $N(n) \rightarrow N(m)$ dependencies are allowed.

Other potential dependencies may be legal when comprehended as part of a specific usage model. For example, these cases:

$P(m) \rightarrow P(n), N(m) \rightarrow N(n), P(m) \rightarrow N(n), N(m) \rightarrow P(n)$ where $m \neq n$

might exist where a device services incoming requests by issuing modified versions of those requests using a different Traffic Class (for example, remapping the first requests address and generating the new request with the resulting address). In these cases, suitable rules must be applied to prevent circular dependencies that would lead to deadlock or livelock.

Examples of devices that may find the above mappings useful:

- Bridges to complex protocols that require state to be save/restored to/from host memory, i.e., PCI Express to Infiniband bridges.
- Messaging engines that must do address translation based upon page tables stored in host memory.
- UMA graphics devices that store their frame buffer in host memory.

Base 6.4 vs Base 6.3

E. ID-Based Ordering Usage §

E.1 Introduction §

ID-Based Ordering (IDO) is a mechanism that permits certain ordering restrictions to be relaxed as a means to improve performance. IDO permits certain TLPs to pass other TLPs in cases where otherwise such passing would be forbidden. The passing permitted by IDO is not required for proper operation (e.g., deadlock avoidance); it is only a means of improving performance.

For discussing IDO, it's useful to introduce the concept of a "TLP stream", which is a set of TLPs that all have the same originator.²²³ For several important cases where TLP passing is normally forbidden, IDO permits such passing to occur if the TLPs belong to different TLP streams.

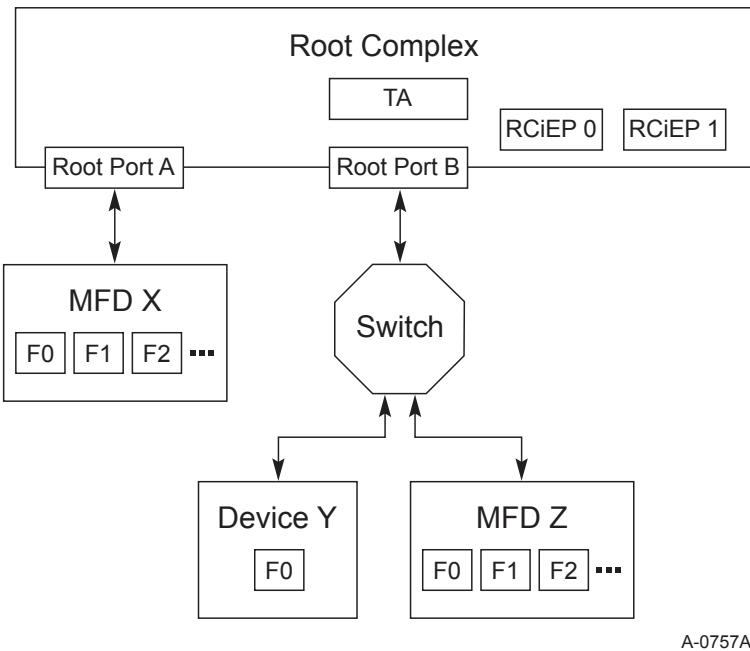


Figure E-1 Reference Topology for IDO Use §

§ Figure E-1 shows a reference topology. The reference topology is not intended to discourage the use of IDO with other topologies, but rather to provide specific examples for discussion.

Devices X and Z are Multi-Function Devices (MFDs); device Y is a Single-Function Device. The RCiEPs are Root Complex Integrated Endpoint Functions, and might or might not be part of the same Device. We will assume that one or more Functions are using the Translation Agent (TA) in the Root Complex (RC).

Referring to the ordering table and descriptions in § Section 2.4.1, having the IDO bit set in a Posted Request, Non-Posted Request, or Completion TLP permits that TLP to pass a Posted Request TLP if the two TLPs belong to different TLP streams. In the following examples, DMAR and DMAW stand for Direct Memory Access Read and Write; PIOR and PIOW stand for Programmed I/O Read and Write.

²²³. That is, the Requester IDs of Requests and the Completer IDs of Completions are all the same.

E.2 Potential Benefits with IDO Use §

Here are some example potential benefits that are envisioned with IDO use. Generally, IDO provides the most benefit when multiple TLP streams share a common Link and that Link becomes congested, either due to high utilization or due to temporary lack of Flow Control (FC) credit.

E.2.1 Benefits for MFD/RP Direct Connect §

Here are some examples in the context of traffic between MFD X and the RC in § Figure E-1 .

- Posted Request passing another Posted Request: when a DMAW from F0 is stalled due to an TA miss, if IDO is set for a DMAW from F1, it is permitted within the RC for this DMAW to pass the stalled DMAW from F0.
- Non-Posted Request passing a Posted Request: when a DMAW from F0 is stalled due to an TA miss, if IDO is set for a DMAR Request from F1, it is permitted within the RC for this DMAR Request to pass the stalled DMAW from F0.
- Completion passing a Posted Request: when a DMAW from F0 is stalled due to an TA miss, if IDO is set for a PIOR Completion from F1, it is permitted within the RC for this PIOR Completion to pass the stalled DMAW from F0.

E.2.2 Benefits for Switched Environments §

Here are some examples in the context of traffic within the Switch in § Figure E-1 .

- Non-Posted Request passing a Posted Request: when a DMAW from Device Y is stalled within the Switch due to a lack of FC credit from Root Port B, if IDO is set for a DMAR Request from MFD Z, it is permitted within the Switch for this DMAR Request to pass the stalled DMAW from Device Y. The same also holds for a DMAR Request from one Function in MFD Z passing a stalled DMAW from a different Function in MFD Z.
- Completion passing a Posted Request: when a DMAW from Device Y is stalled within the Switch due to a lack of FC credit from Root Port B, if IDO is set for a PIOR Completion from MFD Z, it is permitted within the Switch for this PIOR Completion to pass the stalled DMAW from Device Y. The same also holds for a PIOR Completion from one Function in MFD Z passing a stalled DMAW from a different Function in MFD Z.
- Posted Request passing another Posted Request: within a Switch, there is little or no envisioned benefit from having a DMAW from one TLP stream passing a DMAW from a different TLP stream. However, it is not prohibited for Switches to implement such passing as permitted by IDO.

E.2.3 Benefits for Integrated Endpoints §

Here are some examples for the Root Complex Integrated Endpoints (RCiEPs) in § Figure E-1 . The benefits are basically the same as for the MFD/RP Direct Connect case.

- Posted Request passing another Posted Request: when a DMAW from RCiEP 0 is stalled due to an TA miss, if IDO is set for a DMAW from RCiEP 1, it is permitted for this DMAW to pass the stalled DMAW from RCiEP 0.
- Non-Posted Request passing a Posted Request: when a DMAW from RCiEP 0 is stalled due to an TA miss, if IDO is set for a DMAR Request from RCiEP 1, it is permitted for this DMAR Request to pass the stalled DMAW from RCiEP 0.

- Completion passing a Posted Request: when a DMAW from RCiEP 0 is stalled due to an TA miss, if IDO is set for a PIOR Completion from RCiEP 1, it is permitted for this PIOR Completion to pass the stalled DMAW from RCiEP 0.

E.2.4 IDO Use in Conjunction with RO §

IDO and RO²²⁴ are orthogonal. Certain instances of passing; for example, a Posted Request passing another Posted Request, might be permitted by IDO, RO, or both at the same time. While IDO and RO have significant overlap for some cases, it is strongly recommended that both be used whenever safely possible. RO permits certain TLP passing within the same TLP stream, which is never permitted by IDO. For traffic in different TLP streams, IDO permits control traffic to pass any other traffic, and generally it is not safe to Set RO with control traffic.

E.3 When to Use IDO §

With Endpoint Functions²²⁵, it is safe to Set IDO in all applicable TLPs originated by the Endpoint when the Endpoint is directly communicating with only one other entity, most commonly the RC. For the RC case, “directly communicating” specifically includes DMA traffic, PIO traffic, and interrupt traffic; communicating with RCiEPs or communicating using P2P Root Port traffic constitutes communicating with multiple entities.

With a Root Port, there are no envisioned high-benefit use models where it is safe to Set IDO in all applicable TLPs that it originates. Use models where a Root Port Sets IDO in a subset of the applicable TLPs it originates are outside the scope of this specification.

E.4 When Not to Use IDO §

E.4.1 When Not to Use IDO with Endpoints §

With Endpoint Functions, it is not always safe to Set IDO in applicable TLPs it originates if the Endpoint directly communicates with multiple entities. It may be safe to Set IDO in some TLPs and not others, but such use models are outside the scope of this specification.

For example, in § Figure E-1 if Device Y and MFD Z are communicating with P2P traffic and also communicating via host memory, it is not always safe for them to Set IDO in the TLPs they originate. As an example failure case, let’s assume that Device Y does a DMAW (to host memory) followed by a P2P Write to MFD Z. Upon observing the P2P Write, let’s assume that MFD Z then does a DMAW to the same location earlier targeted by the DMAW from Device Y. Normal ordering rules would guarantee that the DMAW from Device Y would be observed by host memory before the DMAW from MFD Z. However, if IDO is set in the DMAW from MFD Z, the RC would be permitted to have the second DMAW pass the first, causing a different end result in host memory contents.

Synchronization techniques like performing zero-length Reads might be used to avoid such communication failures when IDO is used, but specific use models are outside the scope of this specification.

²²⁴. In this Appendix, “RO” is an abbreviation for the Relaxed Ordering Attribute field.

²²⁵. Endpoint Functions include PCI Express Endpoints, Legacy PCI Express Endpoints, and Root Complex Integrated Endpoints.

E.4.2 When Not to Use IDO with Root Ports §

With Root Ports, it is not always safe to Set IDO in applicable TLPs it originates if Endpoint Functions in the hierarchy do any P2P traffic. It may be safe to Set IDO in some TLPs and not others, but such use models are outside the scope of this specification.

As an example, in § Figure E-1 if Device Y and MFD Z are communicating with P2P traffic and also communicating with host software, it is not always safe for Root Port B to Set IDO in the TLPs it originates. For example, let's assume that Device Y does a P2P Write to MFD Z followed by a DMAW (to host memory). Upon observing the DMAW, let's assume that the host does a PIOW to MFD Z. Normal ordering rules would guarantee that the P2P Write from Device Y would be observed by MFD Z before the PIOW from the host. However, if IDO is set in the PIOW from the host, the Switch would be permitted to have the PIOW pass the P2P Write, ultimately having the two Writes arrive at MFD Z out of order.

IMPLEMENTATION NOTE: REQUESTER AND COMPLETER IDS FOR RC-ORIGINATED TLPS §

With RC implementations where the Requester ID in a PIO Request does not match the Completer ID in a DMAR Completion, this enables another potential communication failure case if IDO is Set in the Completion. For this case, if a PIOW is followed by a DMAR Completion with IDO Set, a Switch below the Root Port could permit the DMAR Completion to pass the PIOW, violating the normal ordering rule that a non-RO Read Completion must not pass Posted Requests. The PIOW and DMAR Completion would appear to belong to different TLP streams, though logically they belong to the same TLP stream. Special caution is advised in setting IDO with TLPs originating from such RCs.

E.5 Software Control of IDO Use §

E.5.1 Software Control of Endpoint IDO Use §

By default, Endpoints are not enabled to Set IDO in any TLPs they originate.

IMPLEMENTATION NOTE: THE “SIMPLE” POLICY FOR IDO USE §

It is envisioned that Endpoints designed primarily to communicate directly with only one other entity (e.g., the RC) may find a “simple” policy for setting IDO to be adequate. Here's the envisioned “simple” policy. If the IDO Request Enable bit is Set, the Endpoint Sets IDO in all applicable Request TLPs that it originates. If the IDO Completion Enable bit is Set, the Endpoint Sets IDO in all Completion TLPs that it originates.

It is envisioned that a software driver associated with each Endpoint will determine when it is safe for the Endpoint to set IDO in applicable TLPs it originates. A driver should be able to determine if the Endpoint is communicating with multiple other entities, and should know the Endpoint's capabilities as far as setting IDO with all applicable TLPs when enabled, versus setting IDO selectively. If a driver determines that it is safe to enable the setting of IDO, the driver can set the IDO Request Enable and/or IDO Completion Enable bits either indirectly via OS services or directly, subject to OS policy.

If an Endpoint is designed for communication models where it is not safe to utilize the “simple” policy for IDO use, the Endpoint can implement more complex policies for determining when the Endpoint sets the IDO bit. Such implementations might utilize device-specific controls that are managed by the device driver. Such policies and device-specific control mechanisms are outside the scope of this specification.

E.5.2 Software Control of Root Port IDO Use §

Since there are no envisioned high-benefit “simple” use models for Root Ports setting the IDO bit with TLPs they originate, and there are known communication failure cases if Root Ports set the IDO bit with all applicable TLPs they originate, it is anticipated that Root Ports will rarely be enabled to set IDO in TLPs they originate. Such use models and policies for Root Ports setting IDO are outside the scope of this specification.

Base 6.4 vs Base 6.3

F. Message Code Usage §

§ Table F-1 contains a list of currently defined PCI Express Message Codes. Message codes are defined in this specification and in other specifications. This table will be updated as Messages are defined in other specifications but due to document release schedules, this table might not contain recently defined Messages.

Table F-1 Message Code Usage §

| Message Code | Routing r[2:0] | Type | Description |
|--------------|----------------|------|---|
| 0000 0000 | 011 | Msg | Unlock, see § Section 2.2.8.4 |
| 0000 0001 | 010 | MsgD | Invalidate Request Message, see § Section 10.3.1 |
| 0000 0010 | 010 | Msg | Invalidate Completion Message, see § Section 10.3.2 |
| 0000 0100 | 000 | Msg | Page Request Message, see § Section 10.4.1 |
| 0000 0101 | 010 | Msg | PRG Response Message, see § Section 10.4.2 |
| 0001 0000 | 100 | Msg | Latency Tolerance Reporting (LTR) Message, see § Section 2.2.8.8 |
| 0001 0010 | 100 | Msg | Optimized Buffer Flush/Fill (OBFF) Message, see § Section 2.2.8.9 |
| 0001 0100 | 100 | Msg | PM_Active_State_Nak , see § Section 2.2.8.2 |
| 0001 1000 | 000 | Msg | PM_PME , see § Section 2.2.8.2 |
| 0001 1001 | 011 | Msg | PME_Turn_Off , see § Section 2.2.8.2 |
| 0001 1011 | 101 | Msg | PME_TO_Ack , see § Section 2.2.8.2 |
| 0010 0000 | 100 | Msg | Assert_INTA, see § Section 2.2.8.1 |
| 0010 0001 | 100 | Msg | Assert_INTB, see § Section 2.2.8.1 |
| 0010 0010 | 100 | Msg | Assert_INTC, see § Section 2.2.8.1 |
| 0010 0011 | 100 | Msg | Assert_INTD, see § Section 2.2.8.1 |
| 0010 0100 | 100 | Msg | Deassert_INTA, see § Section 2.2.8.1 |
| 0010 0101 | 100 | Msg | Deassert_INTB, see § Section 2.2.8.1 |
| 0010 0110 | 100 | Msg | Deassert_INTC, see § Section 2.2.8.1 |
| 0010 0111 | 100 | Msg | Deassert_INTD, see § Section 2.2.8.1 |
| 0011 0000 | 000 | Msg | ERR_COR , see § Section 2.2.8.3 |
| 0011 0001 | 000 | Msg | ERR_NONFATAL , see § Section 2.2.8.3 |
| 0011 0011 | 000 | Msg | ERR_FATAL , see § Section 2.2.8.3 |
| 0100 0000 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |
| 0100 0001 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |
| 0100 0011 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |

| Message Code | Routing r[2:0] | Type | Description |
|--------------|-----------------------|----------|--|
| 0100 0100 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |
| 0100 0101 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |
| 0100 0111 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |
| 0100 1000 | 100 | Msg | Ignored Message, see § Section 2.2.8.7 |
| 0101 0000 | 100 | MsgD | Set_Slot_Power_Limit, see § Section 2.2.8.5 |
| 0101 0010 | 100 | Msg | PTM Request, see § Section 2.2.8.10 |
| 0101 0011 | 100 | Msg/MsgD | PTM Response/PTM ResponseD, see § Section 2.2.8.10 |
| 0111 1110 | 000, 010, 011, or 100 | Msg/MsgD | Vendor-Defined Type 0, see § Section 2.2.8.6 |
| 0111 1111 | 000, 010, 011, or 100 | Msg/MsgD | Vendor-Defined Type 1, see § Section 2.2.8.6 |
| 0101 0100 | 010 / 100 | Msg | IDE Sync (see § Section 2.2.8.11) |
| 0101 0101 | 010 / 100 | Msg | IDE Fail (see § Section 2.2.8.11) |

§ Table F-2 contains a list of currently defined Subtype codes for PCI-SIG-Defined VDMs (see § [Section 2.2.8.6.1](#)). Subtype codes are defined in this specification and in other specifications. This table will be updated as Subtype codes are defined in other specifications but due to document release schedules, this table might not contain recently defined Subtypes.

Table F-2 PCI-SIG-Defined VDM Subtype Usage §

| Subtype | Routing r[2:0] | Type | Description |
|-----------|----------------|------|--|
| 0000 0000 | 010 or 011 | MsgD | Deprecated; formerly used for LN Message (Lightweight Notification) |
| 0000 0001 | 011 | MsgD | Hierarchy ID Message, See § Section 2.2.8.6.4 and § Section 6.25 |
| 0000 1000 | 100 | Msg | Device Readiness Status, see § Section 2.2.8.6.2 |
| 0000 1001 | 000 | Msg | Function Readiness Status, see § Section 2.2.8.6.3 |

G. Protocol Multiplexing §

The Protocol Multiplexing mechanism provides a standard mechanism to transport non-PCI Express protocols across a PCI Express Link. The mechanism supports the multiplexing of PMUX Packets and TLPs onto a single PCI Express Link.

An example system topology using Protocol Multiplexing is shown in § Figure G-1 . In this example, the Link may operate in two modes:

- PCI Express Link. Protocol Multiplexing is disabled.
- PMUX Link. Protocol Multiplexing is enabled. Both TLPs and PMUX Packets are used in a coordinated fashion. PMUX Packets may be used to support additional protocols efficiently.

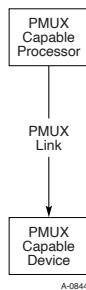


Figure G-1 Device and Processor Connected Using a PMUX Link §

A PMUX Link is shown in § Figure G-2 . Arbitration and encapsulation occurs between the transmit queues and the Link. Demultiplexing and decapsulation occurs between the Link and the various receive queues. Packets are sent from transmit queues to the corresponding receive queues. Packets are identified as either PMUX Packets or TLPs.

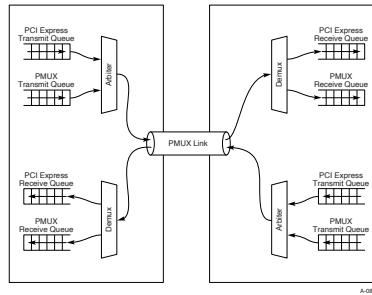


Figure G-2 PMUX Link §

Important attributes of the Protocol Multiplexing mechanism are:

- Protocol Multiplexing support is optional normative.
- Protocol Multiplexing is only supported in Non-Flit Mode. Protocol Multiplexing is not supported in Flit Mode.
- Protocol Multiplexing has no impact on PCI Express components that do not support it.
- Protocol Multiplexing has no impact on PCI Express TLPs and DLLPs, even when it is enabled.

- A Link may be used for both TLPs and PMUX Packets at the same time.
- Protocol Multiplexing does not consume or interfere with PCI Express resources (sequence numbers, credits, etc.). PMUX Packets use distinct resources associated with the specific multiplexed protocol.
- Protocol Multiplexing is disabled by default and is enabled by software. PMUX Packets must not be sent unless enabled by software. PMUX Packets received at Ports that support Protocol Multiplexing are ignored until Protocol Multiplexing is enabled by software.
- Protocol Multiplexing is selectable on a per-Link basis. Protocol Multiplexing may be used on any collection of Links in a system.
- A PMUX Link may support up to four simultaneously active PMUX Channels. Software configures the protocol used on each PMUX Channel.
- PMUX Packets contain an LCRC. This is used to provide data resiliency in a similar fashion as PCI Express TLPs.
- PMUX Packets do not use the ACK/NAK mechanism of PCI Express. Multiplexed protocol specific acknowledgement mechanisms can be used to provide reliable delivery when needed.
- PMUX Packets do not contain a TLP Sequence Number. Instead, they contain a 12 bit PMUX Packet Metadata field that is available for multiplexed protocol specific use.
- PMUX Packet transmitters must contain some arbitration/QoS mechanism for scheduling sending of PMUX Packets, TLPs and DLLPs; however, the mechanism used is outside the scope of this specification.
- The Protocol Multiplexing mechanism does not define any addressing or routing mechanism for PMUX Packets.

PMUX Packets are similar to PCI Express TLPs. The PMUX Packet Flow Through is shown in § Figure G-3 . The PCI Express Packet flow through the layers is shown in § Figure 1-5 . Changes from PCI Express Packet Flow Through are:

- PMUX Packets use a protocol specific PMUX Protocol Layer instead of the PCI Express Transaction Layer.
- PMUX Packets use a simplified Data Link Layer. The packet integrity portion of the Data Link Layer is mostly unchanged (LCRC computation uses a different seed value). The reliability and flow control aspects of the Data Link Layer are removed (the TLP Sequence Number field is repurposed as PMUX Packet Metadata).
- The Physical Layer is slightly modified to provide a mechanism to identify PMUX Packets.

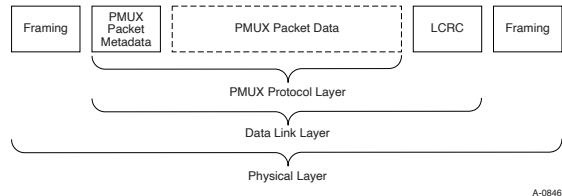


Figure G-3 PMUX Packet Flow Through the Layers §

G.1 Protocol Multiplexing Interactions with PCI Express §

§ Table G-1 and § Table G-2 describe interactions between Protocol Multiplexing and PCI Express. § Table G-1 describes how PCI Express attributes affect Protocol Multiplexing. § Table G-2 describes how Protocol Multiplexing features affect PCI Express attributes.

Table G-1 PCI Express Attribute Impact on Protocol Multiplexing §

| PCI Express Attribute | Impact on Protocol Multiplexing |
|---|--|
| Link Speed | <p>All PMUX Channels are disabled when the Current Link Speed corresponds to a speed that is not supported by Protocol Multiplexing (see § Section G.5.4). A PMUX Channel may be disabled when the Current Link Speed corresponds to a speed that is not supported by the associated protocol.²²⁶</p> <p>Link speed can change either explicitly due to a change in the Target Link Speed field or automatically due to an autonomous Link speed change (see § Section 6.11).</p> <p>The PMUX Protocol Layer is permitted to influence the mechanism used by a component to determine when it requests an autonomous Link speed change. In addition, setting Hardware Autonomous Speed Disable at each end of the Link will prevent certain autonomous Link speed changes (see § Section 7.5.3.18).</p> <p>The PMUX Protocol Layer may be notified of the change.</p> |
| Link Width | <p>A PMUX Channel may be disabled when the Link Width corresponds to a width that is not supported by the associated protocol.²²⁷</p> <p>The PMUX Protocol Layer is permitted to influence the mechanism used by a component to determine when it requests a Link Width change.</p> <p>The PMUX Protocol Layer may be notified of the change.</p> |
| FLR initiated | All PMUX Channels on a Link are disabled if an FLR is directed to the Upstream Port's Function 0. No PMUX Channels are affected if an FLR is directed to any other Function. |
| DL_Down | All PMUX Channels on a Link are disabled. |
| Hot Reset | All PMUX Channels on a Link are disabled. |
| PERST# | All PMUX Channels on a Link are disabled. |
| TLP Replay | No effect on Protocol Multiplexing. |
| DLLP Lost | No effect on Protocol Multiplexing. |
| TLP Prefix | No effect on Protocol Multiplexing. |
| Locked Transactions | No effect on Protocol Multiplexing (including no effect on any protocol specific forwarding of PMUX Packets). |
| AtomicOp Transactions | No effect on Protocol Multiplexing. |
| Multicast Transactions | No effect on Protocol Multiplexing. |
| Access Control Services (ACS) | No effect on Protocol Multiplexing. |
| Alternative Routing ID-Interpretation (ARI) | No effect on Protocol Multiplexing. |
| TPH Requester | No effect on Protocol Multiplexing. |
| Virtual Channels | No effect on Protocol Multiplexing. PCI Express Links remain capable of supporting a full complement of VCs. |

226. The mechanism software uses to determine what Link Speeds are supported by a protocol is outside the scope of this specification.

227. The mechanism software uses to determine what Link Widths are supported by a protocol is outside the scope of this specification.

Base 6.4 vs Base 6.3

| PCI Express Attribute | Impact on Protocol Multiplexing |
|--|--|
| Internal Error | Corrected or Uncorrectable Internal Errors in the PMUX Protocol Layer may be reported as PCI Express Internal Errors. |
| L0s Link Power State | Protocol Multiplexing tracks the Link state. The PMUX Protocol Layer may request the Link transition back to L0. |
| L1 Link Power State | Protocol Multiplexing tracks the Link state. The PMUX Protocol Layer may request the Link transition back to L0. |
| Disabled LTSSM State | Disabling a Link also disables all PMUX Channels on the Link. |
| Loopback LTSSM State | Entering Loopback state disables all PMUX Channels on the Link. |
| Recovery LTSSM State | No effect on Protocol Multiplexing. The PMUX Protocol Layer may be notified. |
| Receiver or Framing Error | The error is reported to the PMUX Protocol Layer to indicate that data might have been lost. This can be used to initiate protocol specific error recovery mechanisms. The Error is reported to software using PCI Express Mechanisms. |
| Lane Reversal | No effect on Protocol Multiplexing. Support for Lane Reversal remains optional. |
| Polarity Inversion | No effect on Protocol Multiplexing. |
| Crosslink | No effect on Protocol Multiplexing. Support for Crosslink remains optional. If supported, the PMUX Protocol Layer may be notified of the outcome of the Crosslink Upstream / Downstream negotiation. |
| Lane assignment rules | Placement and frequency rules for STP Symbols and STP Tokens are not changed (see § Section 4.2.1.2 and § Section 4.2.2.3.2). These rules apply identically to PCI Express TLPs and PMUX Packets. |
| PCI Power Management Power State | All PMUX Channels on a Link are disabled if the Upstream Port's Function 0 is sent to non-D0 state. |
| Dynamic Power Allocation (DPA) | No effect on Protocol Multiplexing. The PMUX Protocol Layer is notified of the change and may participate in the power reduction. PCI Express power management includes any power used by the PMUX Protocol Layer. |
| PCI Power Management Power Consumed / Power Dissipated / Aux_Current | Power required by the PMUX Protocol Layer is included in the PCI structures. |
| Power Budgeting | Power required by the PMUX Protocol Layer is included in the PCI Express structures. |
| Slot Power Limit | Slot Power Limit includes power available to PMUX Protocol Layer. |
| ASPM L0s Entry Condition | <p>The definition of Idle is extended to include:</p> <ul style="list-style-type: none"> • No pending PMUX Packets to transmit over the Link. • For PMUX Channels that use protocol specific Flow Control, no credits are available to send PMUX Packets in that PMUX Channel. |
| ASPM L1 Entry Condition | A Link may not enter L1 if PMUX Packets are pending or scheduled to be transmitted. |
| ASPM L0s/L1 Exit Conditions | A Link may be directed to exit L0s or L1 if a component needs to transmit a PMUX Packet. Routing of PMUX Packets through routing elements is outside the scope of this specification; the associated L0s / L1 exit rules are also unspecified. |

| PCI Express Attribute | Impact on Protocol Multiplexing |
|---------------------------|---|
| Bus Renumbering | No effect on Protocol Multiplexing. |
| Hot Plug | No direct effect on Protocol Multiplexing. Note Hot Plug events indirectly affect Data Link State which, in turn affects Protocol Multiplexing. |
| TLP Sequence Number | No effect on Protocol Multiplexing. PMUX Packets do not consume TLP Sequence Numbers. |
| PCI Express Flow Control | No effect on Protocol Multiplexing. PMUX Packets do not consume PCI Express Flow Control credits. Flow Control Update DLLPs must be sent as required by PCI Express. |
| Error Reporting | No direct effect on Protocol Multiplexing. The PMUX Protocol Layer may be notified when an error is signaled or when an error message is received. |
| LCRC Errors in TLPs | No effect on Protocol Multiplexing. |
| Nullified TLPs | No effect on Protocol Multiplexing. |
| VC Arbitration | No effect on Protocol Multiplexing. Arbitration within PCI Express is unaffected by Protocol Multiplexing. |
| Port Arbitration | No effect on Protocol Multiplexing. Arbitration within PCI Express is unaffected by Protocol Multiplexing. |
| Electrical Idle Inference | PMUX Packets count as TLPs for the purpose of inferring Electrical Idle. |
| MR-IOV | Protocol Multiplexing may co-exist with MR-IOV. PMUX Packets are not part of any MR-IOV Virtual Hierarchy. Protocol Multiplexing is controlled using configuration space in the Management VH(s). |

Table G-2 PMUX Attribute Impact on PCI Express §

| PMUX Attribute | Impact on PCI Express |
|--|---|
| PMUX Protocol Error | No effect on PCI Express. |
| LCRC Errors in PMUX Packets | No effect on PCI Express. PMUX Packets with LCRC errors are discarded without triggering PCI Express replay. This error is reported to the PMUX Protocol Layer and can be used to initiate protocol specific error recovery and/or error reporting mechanisms. |
| Link Unreliability | The PMUX Protocol Layer is permitted to influence the mechanism used by a component to determine if it requests an autonomous link speed change. |
| Nullified PMUX Packets | No effect on PCI Express. It is protocol specific whether PMUX Packets within a specific PMUX Channel may be nullified. If supported, PMUX Packets are nullified in the same manner as TLPs (e.g., inverting the LCRC and signaling nullification at the Physical Layer). Receiving a nullified PMUX Packet may be reported to the PMUX Protocol Layer. |
| Electrical Idle Inference | PMUX Packets count as TLPs for the purpose of inferring Electrical Idle. |
| PMUX Protocol Layer directs LTSSM to enter Recovery | Both PCI Express and the PMUX Protocol Layer are permitted to direct a transition from L0 to Recovery. |
| PMUX Channel Enabled / Disabled | No effect on PCI Express |
| PMUX Packet Receiver Buffer Overflow | No effect on PCI Express. This is a protocol problem within the PMUX Channel. The PMUX Transport Layer must continue to accept such packets and dispose of them using protocol specific mechanisms. |
| Received PMUX Packet larger or smaller than supported by the associated protocol | No effect on PCI Express. These are protocol problems within the PMUX Channel. The PMUX Transport Layer must accept such packets and dispose of them using protocol specific mechanisms. |

| PMUX Attribute | Impact on PCI Express |
|---|--|
| Received PMUX Packet that contains more than 125 DWORDs of PMUX Packet Data | <p>No effect on PCI Express. This is an invalid PMUX Packet. The PMUX Transport Layer must accept such a packet and dispose of it. A protocol specific mechanism may be used to report the error.</p> <p>Note: This situation only exists for a packet encoded using 8b/10b. The TLP Length field of a packet encoded using 128b/130b cannot contain values that cause this situation.</p> |
| PMUX Packet Received on disabled PMUX Channel | <p>No effect on PCI Express. No effect on any other PMUX Channel. Receivers must silently ignore such packets regardless of packet length and regardless of whether or not the packet is nullified.</p> <p>PMUX Packets arriving on a disabled PMUX Channel may occur normally when software is in the process of initializing Protocol Multiplexing.</p> |
| PMUX Packet Received at component that does not support Protocol Multiplexing | <p>Software should not enable PMUX Packets unless both ends of a Link support Protocol Multiplexing.</p> <p>In the 128b/130b encoding, receiving a PMUX Packet by a component that does not support Protocol Multiplexing is a Framing Error (see § Section 4.2.2.3.1).</p> <p>In the 8b/10b encoding, the PMUX Packet LCRC is computed differently than the TLP LCRC. Receivers that do not support Protocol Multiplexing will interpret PMUX Packets as TLPs with LCRC errors and will not process them.</p> |
| Large PMUX Packets when PCI Express Max_Payload_Size is small | <p>Under certain conditions, it is possible for a large PMUX Packet to trigger a premature PCI Express replay. For example, this can occur when the time needed to transmit a PMUX Packet is larger than the REPLAY_TIMER (see § <u>Section 3.6.2.1</u>).</p> <p>To avoid this issue, implementations are permitted to not advance (hold) their REPLAY_TIMER during the reception of PMUX Packets.</p> <p>Note: The PCI Express REPLAY_TIMER mechanism has adequate headroom for most cases. This issue exists when (1) Max_Payload_Size is 000b, (2) PMUX Packets are larger than about 80 DWORDs, and (3) the REPLAY_TIMER is at the low end of the -0%/+100% tolerance.</p> |

G.2 PMUX Packets §

A PMUX Packet contains the information shown in § [Figure G-4](#).

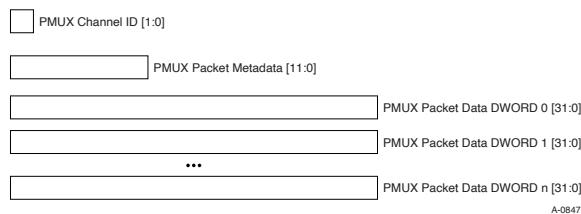


Figure G-4 PMUX Packet §

PMUX Channel ID is a 2 bit field that identifies which protocol is associated with a PMUX Packet. PMUX Channel ID values are between 0 and 3 (inclusive).

PMUX Packet Metadata is a 12 bit field that provides information about the PMUX Packet. Definition of this field is protocol specific and is outside the scope of this specification.

A PMUX Packet consists of between 0 and 125 DWORDs of PMUX Packet Data. Layout and usage of these DWORDs is protocol specific and is outside the scope of this specification. A PMUX Packet need not have any PMUX Packet Data and may consist only of PMUX Channel ID and PMUX Packet Metadata.

G.3 PMUX Packet Layout §

There are two layouts defined for PMUX Packets. One layout is used for 2.5 and 5.0 GT/s data rates and another layout is used for 8.0 GT/s and higher data rates. These layouts are discussed in the following sections.

G.3.1 PMUX Packet Layout for 8b/10b Encoding §

§ Figure G-5 and § Table G-3 show the layout of PMUX Packets when using 8b/10b encoding. For reference, the 8b/10b encoding of a TLP is also shown (see § Section 4.2.1.2 for the official definition).

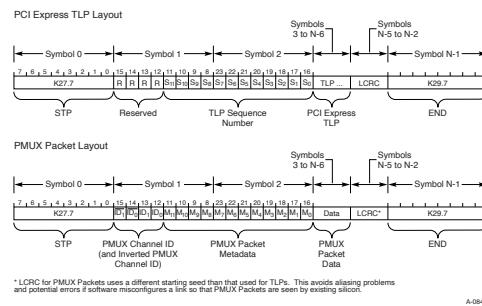


Figure G-5 TLP and PMUX Packet Framing (8b/10b Encoding) §

Table G-3 PMUX Packet Layout (8b/10b Encoding) §

| Symbol | Field | Bit Position(s) | PMUX Packet Usage | TLP Usage |
|------------|-------------------------------|-----------------|---|---------------------------|
| 0 | Start TLP Indicator | 7:0 | K27.7 | |
| 1 | Inverted PMUX Channel ID[1:0] | 7:6 | Inverted (1s complement) of Symbol 1 bits 6:4 | Reserved |
| | PMUX Channel ID[1:0] | 5:4 | PMUX Channel ID | Reserved |
| | PMUX Packet Metadata[11:8] | 3:0 | PMUX Packet Metadata[11:8] | TLP Sequence Number[11:8] |
| 2 | PMUX Packet Metadata[7:0] | 7:0 | PMUX Packet Metadata[7:0] | TLP Sequence Number[7:0] |
| 3 to N-6 | Packet | 7:0 | PMUX Packet | TLP |
| N-5 to N-2 | LCRC | 7:0 | PMUX LCRC | PCI Express LCRC |
| N-1 | END | 7:0 | K29.7 | |

For PMUX Packets, symbols 1 and 2 contain PMUX Packet Metadata in the same bit positions that TLPs use for TLP Sequence Number.

The PMUX LCRC algorithm is identical to the TLP LCRC algorithm as described in [§ Section 3.6.2](#) with the following modifications:

- The seed value is FB3E E248h (TLP LCRC uses FFFF FFFFh).
- The PMUX Channel ID field in Symbol 1 bits 7:4 is included in the PMUX LCRC in the same manner as the 4 reserved bits in the TLP LCRC.
- The PMUX Packet Metadata field is included in the PMUX LCRC in the same manner as the TLP Sequence Number field is included in the TLP LCRC.

IMPLEMENTATION NOTE: PMUX PACKETS AT RECEIVERS THAT DO NOT SUPPORT PROTOCOL MULTIPLEXING §

The bits used for PMUX Channel ID are reserved unless Protocol Multiplexing is supported. As such, Receivers that do not support Protocol Multiplexing must ignore the PMUX Channel ID bits. If software misconfigures Protocol Multiplexing, a component that does not support Protocol Multiplexing could receive a PMUX Packet. To prevent that component from misinterpreting such a PMUX Packet as a valid TLP, the LCRC computation is changed for PMUX Packets. The result is that a valid PMUX Packet will never be misinterpreted as a valid TLP. These LCRC “errors” may trigger PCI Express replay and may result in REPLAY_NUM Rollover correctable errors being reported.

IMPLEMENTATION NOTE: PMUX PACKET LCRC §

The PMUX Channel ID field is covered by the LCRC. As such, when using 8b/10b encoding, receivers must wait until the LCRC is checked to make firm decisions based on the PMUX Channel ID value. The Inverted PMUX Channel ID can be compared against the PMUX Channel ID to make tentative decisions.

Note: The value of the LCRC associated with a given PMUX Packet is independent of the encoding used to transmit the packet.

G.3.2 PMUX Packet Layout at 128b/130b Encoding §

§ Figure G-6 and § Table G-4 show the layout of PMUX Packets when using 128b/130b encoding. For reference, the 128b/130b encoding of a TLP is also shown (see § Section 4.2.2.2 for the official definition).

Base 6.4 vs Base 6.3

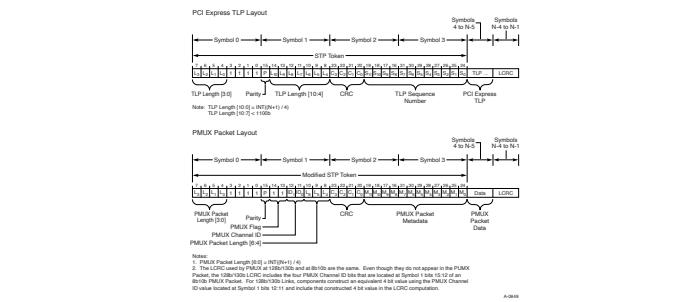


Figure G-6 TLP and PMUX Packet Framing (128b/130b Encoding) §

Table G-4 PMUX Packet Layout (128b/130b Encoding) §

| Symbol | Field | Bit Position(s) | PMUX Packet Usage | TLP Usage |
|------------|----------------------------|-----------------|--|---|
| 0 | Start TLP Indicator | 3:0 | Value of 1111b | |
| | PMUX Packet Length[3:0] | 7:4 | Bits [3:0] of the PMUX Packet Length. Bit 0 is the least significant PMUX Packet Length bit. | Bits [3:0] of the TLP Length field. Bit 0 is the least significant TLP Length bit. |
| 1 | Frame Parity (P) | 7 | Even parity of Symbol 0 bits [7:4], Symbol 1 bits [6:0] and Symbol 2 bits [7:4] | |
| | PMUX Packet Indicator | 6:5 | Value of 11b | |
| | PMUX Channel ID[1:0] | 4:3 | PMUX Channel ID | |
| 2 | PMUX Packet Length[6:4] | 2:0 | Bits [6:4] of PMUX Packet Length. Bit 6 is the most significant PMUX Packet Length bit. | Bits [10:4] of the TLP Length field. Bit 10 is the most significant TLP Length bit. |
| | PMUX Packet Metadata[11:8] | 3:0 | PMUX Packet Metadata[11:8] | |
| | Frame CRC (C[3:0]) | 7:4 | CRC of Symbol 0, bits [7:4] and Symbol 1 bits [6:0] | |
| 3 | PMUX Packet Metadata[7:0] | 7:0 | PMUX Packet Metadata[7:0] | TLP Sequence Number[7:0] |
| 4 to N-5 | Packet | 7:0 | PMUX Packet | TLP |
| N-4 to N-1 | LCRC | 7:0 | PMUX LCRC | PCI Express LCRC |

§ Table G-5 describes the encodings of Symbol 1 bits [6:3] in more detail. If these bits contain a value less than 1001b, the packet is a TLP and is processed as described in § Section 4.2.2.²²⁸ If these bits contain 1001b, 1010b, or 1011b, the encoding is reserved for future standardization and is processed as described in § Section 4.2.2.3.3. If these bits contain a value greater than or equal to 1100b, the packet is a PMUX Packet is defined as specified in this appendix.²²⁹

228. The value 1001b supports a maximum TLP Length [10:0] value of 1151 DWORDS (decimal). This will accommodate a TLP consisting of 4096 bytes of payload, 16 bytes of TLP Header, 4 bytes of TLP digest, and 480 bytes of TLP Prefix.

Table G-5 Symbol 1 Bits [6:3] §

| Symbol 1 bits [6:3] | Meaning |
|------------------------|---|
| 0xxxb or 1000b | Packet is a TLP. Bits [6:3] are TLP Length [10:7]. |
| 1001b, 1010b, or 1011b | Encoding reserved for future standardization. Receivers detecting these encodings shall process them as described in § Section 4.2.2.3.3. |
| 1100b | Packet is a PMUX Packet. PMUX Channel ID is 0. |
| 1101b | Packet is a PMUX Packet. PMUX Channel ID is 1. |
| 1110b | Packet is a PMUX Packet. PMUX Channel ID is 2. |
| 1111b | Packet is a PMUX Packet. PMUX Channel ID is 3. |

For PMUX Packets, the packet length in DWORDs is contained in PMUX Packet Length [6:0]. Other than being a smaller field, PMUX Packet Length is interpreted in the same manner as TLP Length. Specifically, PMUX Packet Length also includes the framing and PMUX LCRC DWORDs (see § Section 4.2.2.2).

For PMUX Packets, symbols 2 and 3 contain PMUX Packet Metadata in the same bit positions that TLPs use for TLP Sequence Number.

The PMUX LCRC algorithm is identical to the TLP LCRC algorithm as described in [§ Section 3.6.2](#) with the following modifications:

- The seed value is FB3E E248h (TLP LCRC uses FFFF FFFFh).
- The PMUX Channel ID field in Symbol 1 bits 4:3 is used to compute a 4 bit value that is included in the PMUX LCRC in the same manner as the 4 reserved bits in the TLP LCRC. This 4 bit value contains the value that would be used, by the 8b/10b encoding, for Symbol 1 bits 7:4. Specifically, the lower 2 bits of this 4 bit value contain the PMUX Channel ID and the upper 2 bits contain the inverse (1s complement) of the PMUX Channel ID.
- The PMUX Packet Metadata field is included in the PMUX LCRC in the same manner as the TLP Sequence Number field is included in the TLP LCRC.

The Frame CRC and Frame Parity fields are computed as shown below. This is the same algorithm computed over the same bit positions as defined in § Section 4.2.2.2 .

$$\begin{aligned}
 C[0] &= 1b \wedge \text{PMUX_Channel_ID}[0] \\
 &\quad \wedge L[6] \wedge L[4] \wedge L[2] \wedge L[1] \wedge L[0] \\
 C[1] &= 1b \wedge 1b \wedge \text{PMUX_Channel_ID}[0] \\
 &\quad \wedge L[5] \wedge L[4] \wedge L[3] \wedge L[2] \\
 C[2] &= 1b \wedge \text{PMUX_Channel_ID}[1] \\
 &\quad \wedge L[6] \wedge L[4] \wedge L[3] \wedge L[2] \wedge L[1] \\
 C[3] &= \text{PMUX_Channel_ID}[1] \\
 &\quad \wedge \text{PMUX_Channel_ID}[0] \\
 &\quad \wedge L[5] \wedge L[3] \wedge L[2] \wedge L[1] \wedge L[0] \\
 P &= 1b \wedge 1b \wedge \text{PMUX_Channel_ID}[1] \\
 &\quad \wedge \text{PMUX_Channel_ID}[0] \\
 &\quad \wedge L[6] \wedge L[5] \wedge L[4] \wedge L[3] \wedge L[2] \wedge L[1] \wedge L[0] \\
 &\quad \wedge C[3] \wedge C[2] \wedge C[1] \wedge C[0]
 \end{aligned}$$

229. The value 1100b was chosen to simplify distinguishing PMUX Packets from TLPs and from the reserved encodings.

IMPLEMENTATION NOTE: PMUX CHANNEL ID AND FRAME CRC §

When using 128b/130b encoding, the PMUX Channel ID field is covered by the Frame CRC and Frame Parity fields. As such, receivers may make decisions based on the PMUX Channel ID value as soon as the Frame CRC and Frame Parity is checked and need not wait until the PMUX LCRC is checked.

Note: The PMUX Channel ID is also covered by the LCRC. The value of the LCRC associated with a given PMUX Packet is independent of the encoding used to transmit the packet.

G.4 PMUX Control §

Protocol Multiplexing is disabled by default. Each PMUX Channel must be explicitly enabled by software at each end of the associated Link. Protocol Multiplexing is disabled whenever the link drops (Data Link Layer indicates DL_Down).

A component that supports Protocol Multiplexing indicates such by the presence of the PMUX Extended Capability.

The following rules apply to components that support Protocol Multiplexing:

- PMUX Packets received in a PMUX Channel that is not enabled are silently ignored.
- PMUX Packets may not be transmitted unless the associated PMUX Channel is enabled. A PMUX Channel may also require additional, protocol specific, initialization mechanisms before PMUX Packets may be transmitted.

G.5 PMUX Extended Capability §

§ Figure G-7 shows the PMUX Extended Capability structure. The presence of this capability indicates that the Port supports the optional Protocol Multiplexing mechanism. This capability is optional and may be present in any Downstream Port and in Function 0 of any Upstream Port. It must not be present in non-zero Functions of Upstream Ports or in RCRBs.

The length of the PMUX Extended Capability is determined by the PMUX Protocol Array Size field (see § Section G.5.2).

This capability contains a list of the protocols supported by the Link (the PMUX Protocol Array). It also contains the mechanism software uses to enable and configure PMUX Channels. This capability must be present in both the Upstream and Downstream Ports of a Link in order for Protocol Multiplexing to be successfully enabled.

Software may enable the Upstream and Downstream Ports of a Link in either order. Software may enable multiple PMUX Channels using a single write to the PMUX Control Register.

Behavior is undefined if software enables Protocol Multiplexing in one Port and the other Port of the Link does not support Protocol Multiplexing. Behavior is also undefined if software configures a PMUX Channel inconsistently (the same PMUX Channel in the Ports on each end of a Link configured with incompatible protocols).

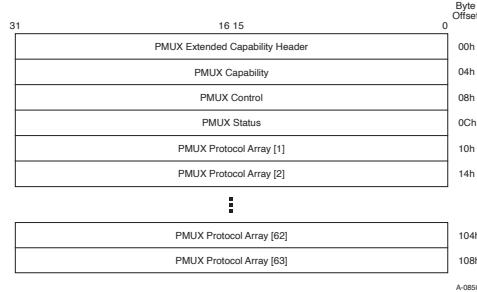


Figure G-7 PMUX Extended Capability Header

G.5.1 PMUX Extended Capability Header (Offset 00h)

§ Figure G-8 details the allocation of fields in the PMUX Extended Capability header ; § Table G-6 provides the respective bit definitions.

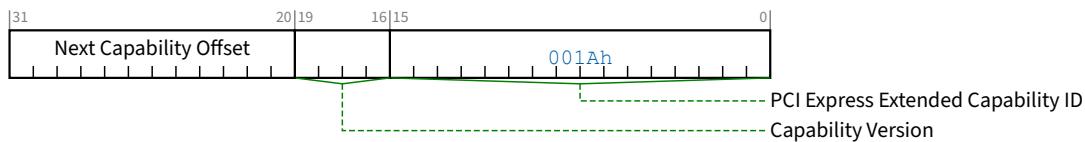


Figure G-8 PMUX Extended Capability Header

Table G-6 PMUX Extended Capability Header

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| 15:0 | PCI Express Extended Capability ID - This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the PMUX Extended Capability is 001Ah . | RO |
| 19:16 | Capability Version - This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | RO |
| 31:20 | Next Capability Offset - This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of capabilities. This offset is relative to the beginning of PCI compatible Configuration Space and thus must always be either 000h (for terminating the list of Capabilities) or greater than OFFh. | RO |

G.5.2 PMUX Capability Register (Offset 04h)

§ Figure G-9 details the allocation of fields in the PMUX Capability Register . § Table G-7 provides the respective bit definitions.

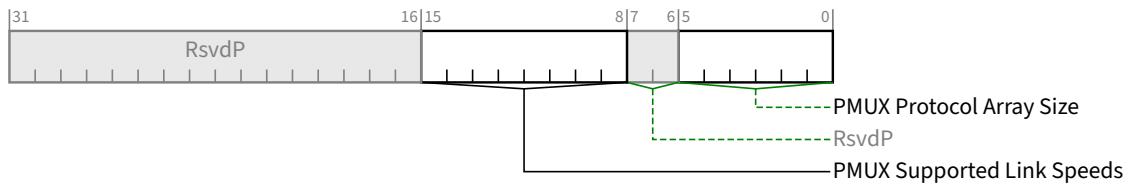


Figure G-9 PMUX Capability Register §

Table G-7 PMUX Capability Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 5:0 | PMUX Protocol Array Size - Indicates the size of this Function's PMUX Protocol Array. This field may be 0 to indicate that even though no protocols are supported, the Port will ignore all received PMUX Packets. | RO |
| 15:8 | PMUX Supported Link Speeds - This field indicates the Link speed(s) where Protocol Multiplexing is supported. Each bit corresponds to a Link speed. If a bit is Set, Protocol Multiplexing is supported at that Link speed. If a bit is Clear, Protocol Multiplexing is not supported at that Link speed. Bit definitions are: Bit 8 2.5 GT/s Bit 9 5.0 GT/s Bit 10 8.0 GT/s Bit 11 16.0 GT/s Bit 12 32.0 GT/s Bits 15:13 RsvdP At least one Link speed must be supported (i.e., the field must be non-zero). A Port may support any combination of Link speeds. For example, this field could contain the value 0000 0100b indicating that Protocol Multiplexing is only supported at 8.0 GT/s. This field must not indicate support for Link speeds that are not supported by the Link (see § Section 7.5.3.18). Note that this field indicates the Link speeds supported by Protocol Multiplexing for the Link. The Link speeds that a particular protocol supports and the mechanism used to report that information are protocol specific. | RO / RsvdP |

G.5.3 PMUX Control Register (Offset 08h) §

§ Figure G-10 details the allocation of fields in the PMUX Control Register. § Table G-8 provides the respective bit definitions.

Channel n is enabled and available for use by the PMUX Protocol Layer when all of the following are true:

- The Channel n Assignment field is non-zero.
- The Channel n Assignment field is less than or equal to PMUX Protocol Array Size.
- The Channel n Assignment field indicates an implemented entry in the PMUX Protocol Array (see § [Section G.5.5](#)).
- All of the PMUX Channel n Disabled bits are Clear (see § [Section G.5.4](#)).

Otherwise, Channel n is disabled.

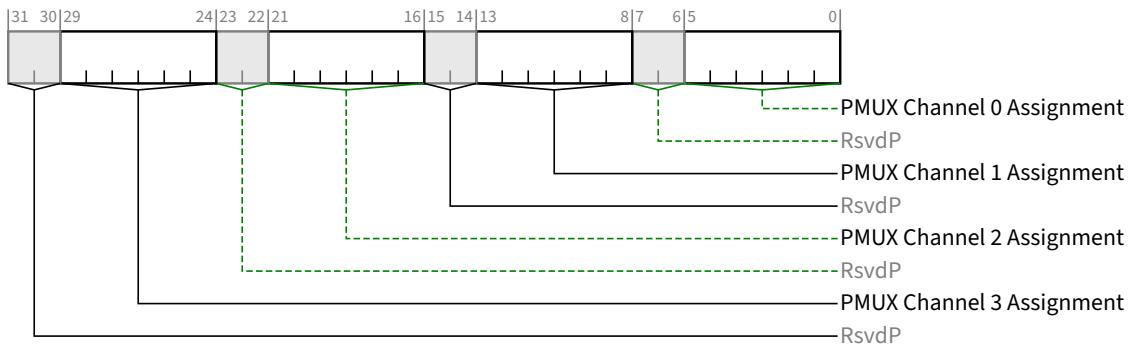


Figure G-10 PMUX Control Register §

Table G-8 PMUX Control Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 5:0 | <p>PMUX Channel 0 Assignment - This field indicates the protocol assigned to PMUX Channel 0. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 0.</p> <p>If PMUX Protocol Array Size is less than 63 (see § Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0.</p> <p>This field defaults to 0h.</p> | RW |
| 13:8 | <p>PMUX Channel 1 Assignment - This field indicates the protocol assigned to PMUX Channel 1. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 1.</p> <p>If PMUX Protocol Array Size is less than 63 (see § Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0.</p> <p>This field defaults to 0h.</p> | RW |
| 21:16 | <p>PMUX Channel 2 Assignment - This field indicates the protocol assigned to PMUX Channel 2. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 2.</p> <p>If PMUX Protocol Array Size is less than 63 (see § Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0.</p> <p>This field defaults to 0h.</p> | RW |
| 29:24 | <p>PMUX Channel 3 Assignment - This field indicates the protocol assigned to PMUX Channel 3. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 3.</p> <p>If PMUX Protocol Array Size is less than 63 (see § Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0.</p> <p>This field defaults to 0h.</p> | RW |

G.5.4 PMUX Status Register (Offset 0Ch) §

§ Figure G-11 details the allocation of fields in the PMUX Status Register. § Table G-9 provides the respective bit definitions.

Each channel has a set of Disabled bits. When Channel n Assignment field is non-zero, the Channel n Disabled bits reflect the error status of the channel. The following Disabled bits are defined:

- PMUX Channel n Disabled: Link Speed
- PMUX Channel n Disabled: Link Width
- PMUX Channel n Disabled: Protocol Specific

When there are multiple reasons for disabling a channel, an implementation may choose which reason(s) to report. For example, if a protocol needs bandwidth equivalent to x1 8.0 GT/s, when there is inadequate bandwidth (e.g., the Link is operating at x1 5.0 GT/s, x1 2.5 GT/s, or x2 2.5 GT/s), it could disable the PMUX Channel by indicating any or all of Disabled: Link Width, Disabled: Link Speed, or Disabled: Protocol Specific.

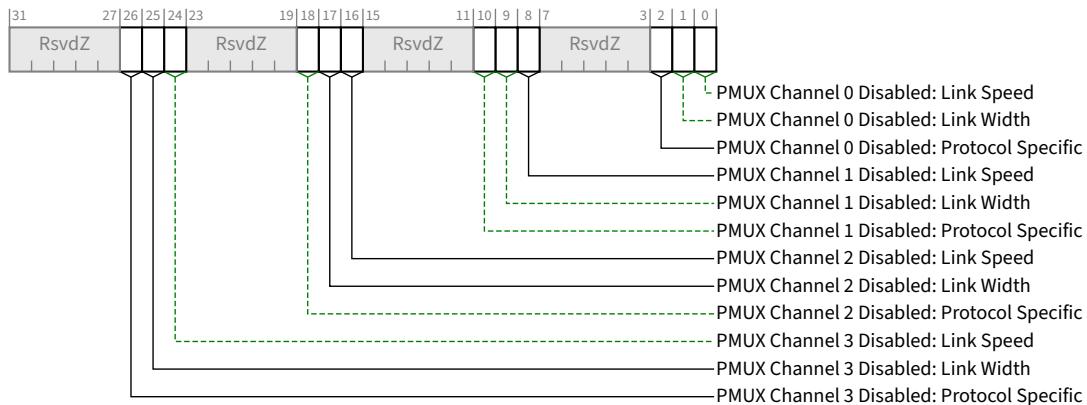


Figure G-11 PMUX Status Register §

Table G-9 PMUX Status Register §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 0 | PMUX Channel 0 Disabled: Link Speed - If Set, Channel 0 is disabled because the Current Link Speed (§ Section 7.8.9) is not supported by Protocol Multiplexing or by the protocol assigned to Channel 0. This bit is 0 when no protocol is assigned to Channel 0 (i.e., Channel 0 Control field is 0h). | RO |
| 1 | PMUX Channel 0 Disabled: Link Width - If Set, Channel 0 is disabled because the current Link Width is not supported by the protocol assigned to Channel 0. This bit is 0 when no protocol is assigned to Channel 0 (i.e., PMUX Channel 0 Assignment field is 0h). | RO |
| 2 | PMUX Channel 0 Disabled: Protocol Specific - If Set, Channel 0 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 0 (i.e., PMUX Channel 0 Assignment field is 0h). | RO |
| 8 | PMUX Channel 1 Disabled: Link Speed - If Set, Channel 1 is disabled because the Current Link Speed (§ Section 7.8.9) is not supported by Protocol Multiplexing or by the protocol assigned to Channel 1. | RO |

| Bit Location | Register Description | Attributes |
|--------------|--|------------|
| | This bit is 0 when no protocol is assigned to Channel 1 (i.e., PMUX Channel 1 Assignment field is 0h). | |
| 9 | PMUX Channel 1 Disabled: Link Width - If Set, Channel 1 is disabled because the current Link Width is not supported by the protocol assigned to Channel 1. This bit is 0 when no protocol is assigned to Channel 1 (i.e., PMUX Channel 1 Assignment field is 0h). | RO |
| 10 | PMUX Channel 1 Disabled: Protocol Specific - If Set, Channel 1 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 1 (i.e., PMUX Channel 1 Assignment field is 0h). | RO |
| 16 | PMUX Channel 2 Disabled: Link Speed - If Set, Channel 2 is disabled because the Current Link Speed (§ Section 7.8.9) is not supported by Protocol Multiplexing or by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 2 (i.e., PMUX Channel 2 Assignment field is 0h). | RO |
| 17 | PMUX Channel 2 Disabled: Link Width - If Set, Channel 2 is disabled because the current Link Width is not supported by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 2 (i.e., PMUX Channel 2 Assignment field is 0h). | RO |
| 18 | PMUX Channel 2 Disabled: Protocol Specific - If Set, Channel 2 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 2 (i.e., PMUX Channel 2 Assignment field is 0h). | RO |
| 24 | PMUX Channel 3 Disabled: Link Speed - If Set, Channel 3 is disabled because the Current Link Speed (§ Section 7.8.9) is not supported by Protocol Multiplexing or by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 3 (i.e., PMUX Channel 3 Assignment field is 0h). | RO |
| 25 | PMUX Channel 3 Disabled: Link Width - If Set, Channel 3 is disabled because the current Link Width is not supported by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 3 (i.e., PMUX Channel 3 Assignment field is 0h). | RO |
| 26 | PMUX Channel 3 Disabled: Protocol Specific - If Set, Channel 3 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 3 (i.e., PMUX Channel 3 Assignment field is 0h). | RO |

G.5.5 PMUX Protocol Array (Offsets 10h through 48h) §

The PMUX Protocol Array consists of up to 63 entries. The size of the PMUX Protocol Array is indicated by the PMUX Protocol Array Size field (see § Section G.5.2).

§ Figure G-12 details the allocation of fields in each PMUX Protocol Array Entry. § Table G-10 provides the respective bit definitions.

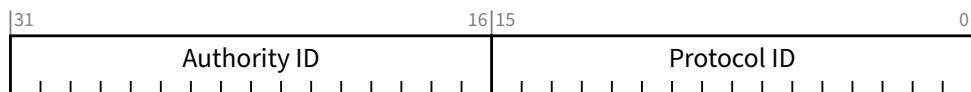


Figure G-12 PMUX Protocol Array Entry §

Table G-10 PMUX Protocol Array Entry §

| Bit Location | Register Description | Attributes |
|--------------|---|------------|
| 15:0 | <p>Protocol ID - In conjunction with Authority ID designates a specific protocol and the mechanism by which that protocol is mapped onto Protocol Multiplexing.</p> <p>This value is assigned by the Vendor associated with the <u>Authority ID</u> field.</p> | RO |
| 31:16 | <p>Authority ID - Designates the authority controlling the values used in the <u>Protocol ID</u> field.</p> <p>The Authority ID field contains a Vendor ID as assigned by the PCI-SIG.</p> | RO |

The value 0000 0000h indicates an unimplemented PMUX Protocol Array Entry . The PMUX Protocol Array is indexed starting at 1.

PMUX Channel n is enabled and configured to support the protocol associated with PMUX Protocol Array Entry at index m when the PMUX Channel n Assignment field contains the value m (see § Section G.5.3).

Entries in the PMUX Protocol Array with the Authority ID value 1 (0001h) represent protocols that are defined by the PCI-SIG.

Duplicate Entries in the PMUX Protocol Array may be used to represent multiple instances of a particular protocol. This permits software control of the mapping between PMUX Channel ID and a specific instance of a protocol.

IMPLEMENTATION NOTE: MULTIPLE PROTOCOL INSTANCES §

A Link may have a single PMUX Protocol assigned to multiple PMUX Channels. Each PMUX Channel is assigned to a different instance of the protocol. Each instance of a protocol corresponds to an entry in the PMUX Protocol Array.

Consider a Port that supports two instances of protocol X. Two entries in the PMUX Protocol Array would indicate protocol X (indexes A and B for example). To assign instance A to PMUX Channel 0 and instance B to PMUX Channel 2, place the value A in the PMUX Channel 0 Assignment field and the value B in the PMUX Channel 2 Assignment field.

Base 6.4 vs Base 6.3

H. Flow Control Update Latency and ACK Update Latency Calculations §

This appendix is informational only and should not be considered normative. Earlier revisions of this specification outlined the method used to calculate the Flow Control Update Latency and Ack Update Latency. This appendix describes the method used to derive the values and preserves the UpdateFactor and AckFactor values.

H.1 Flow Control Update Latency §

Recommended Flow Control Update Latency is described in the Implementation Note in § Section 2.6.1.2 entitled, Non-Flit Mode Flow Control Update Latency.

Flow Control Update Latency tables were simplified in the 4.0 Revision of this specification. At that time, the original tables were moved to this appendix. Note that in the 4.0 Revision of this specification, Tables 2-47 and 2-48 were distinct, but identical tables. The 5.0 Revision contains a single table that applies to 8.0 GT/s or higher.

| Original Tables | | Simplified Tables | |
|-----------------|-------------------|-------------------|-------------------|
| Base 3.1 | Base 4.0 or later | Base 4.0 | Base 5.0 or later |
| Table 2-42 | § Table H-1 | Table 2-45 | § Table 2-52 |
| Table 2-43 | § Table H-2 | Table 2-46 | § Table 2-53 |
| Table 2-44 | § Table H-3 | Table 2-47 | § Table 2-54 |
| n/a | n/a | Table 2-48 | |

The values in the Tables are measured starting from when the Receiver Transaction Layer makes additional receive buffer space available by processing a received TLP, to when the first Symbol of the corresponding UpdateFC DLLP is transmitted. The values are calculated as a function of the largest TLP payload size and Link width using the formula:

$$\frac{(\text{Rx_MPS_Limit} + \text{TLPOverhead}) \times \text{UpdateFactor}}{\text{LinkWidth}} + \text{InternalDelay}$$

Equation H-1 Max UpdateFC Latency §

where:

Rx_MPS_Limit

The Rx_MPS_Limit value as determined in § Section 2.2.2. For a Multi-Function Device where different Functions have different Rx_MPS_Limit values, using the smallest Rx_MPS_Limit value across all Functions is strongly recommended.

TLPOverhead

Represents the additional TLP components which consume Link bandwidth (TLP Prefix, header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.

UpdateFactor

Represents the number of maximum size TLPs sent during the time between UpdateFC receptions, and is used to balance Link bandwidth efficiency and receive buffer sizes - the value varies according to Rx_MPS_Limit and Link width. § Table H-1, § Table H-2, and § Table H-3 below include the UpdateFactor(UF).

LinkWidth

The operating width of the Link

InternalDelay

Represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times for 2.5 GT/s mode operation, 70 Symbol Times for 5.0 GT/s mode operation, and 115 Symbol Times for 8.0 GT/s and 16.0 GT/s modes of operation.

*Table H-1 Maximum UpdateFC Transmission Latency Guidelines for 2.5 GT/s Mode
Operation by Link Width and Max Payload (Symbol Times) §*

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

*Table H-2 Maximum UpdateFC Transmission Latency Guidelines for 5.0 GT/s Mode
Operation by Link Width and Max Payload (Symbol Times) §*

| | | Link Operating Width | | | | | | |
|-------------------------|------|----------------------|------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Rx_MPS_Limit (bytes) | 128 | 288 UF = 1.4 | 179 UF = 1.4 | 124 UF = 1.4 | 118 UF = 2.5 | 109 UF = 3.0 | 99 UF = 3.0 | 84 UF = 3.0 |
| | 256 | 467 UF = 1.4 | 268 UF = 1.4 | 169 UF = 1.4 | 158 UF = 2.5 | 141 UF = 3.0 | 123 UF = 3.0 | 96 UF = 3.0 |
| | 512 | 610 UF = 1.0 | 340 UF = 1.0 | 205 UF = 1.0 | 137 UF = 1.0 | 160 UF = 2.0 | 137 UF = 2.0 | 103 UF = 2.0 |
| | 1024 | 1122 UF = 1.0 | 596 UF = 1.0 | 333 UF = 1.0 | 201 UF = 1.0 | 245 UF = 2.0 | 201 UF = 2.0 | 135 UF = 2.0 |
| | 2048 | 2146 UF = 1.0 | 1108 UF = 1.0 | 589 UF = 1.0 | 329 UF = 1.0 | 416 UF = 2.0 | 329 UF = 2.0 | 199 UF = 2.0 |

| | | Link Operating Width | | | | | | |
|------|--|----------------------|------------------|------------------|-----------------|-----------------|-----------------|-----------------|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| 4096 | | 4194 UF = 1.0 | 2132 UF = 1.0 | 1101 UF = 1.0 | 585 UF = 1.0 | 757 UF = 2.0 | 585 UF = 2.0 | 327 UF = 2.0 |

Table H-3 Maximum UpdateFC Transmission Latency Guidelines for 8.0 GT/s Operation by Link Width and Max Payload (Symbol Times) §

| | | Link Operating Width | | | | | | |
|-------------------------|------|----------------------|------------------|------------------|-----------------|-----------------|-----------------|-----------------|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Rx_MPS_Limit (bytes) | 128 | 333 UF = 1.4 | 224 UF = 1.4 | 169 UF = 1.4 | 163 UF = 2.5 | 154 UF = 3.0 | 144 UF = 3.0 | 129 UF = 3.0 |
| | 256 | 512 UF = 1.4 | 313 UF = 1.4 | 214 UF = 1.4 | 203 UF = 2.5 | 186 UF = 3.0 | 168 UF = 3.0 | 141 UF = 3.0 |
| | 512 | 655 UF = 1.0 | 385 UF = 1.0 | 250 UF = 1.0 | 182 UF = 1.0 | 205 UF = 2.0 | 182 UF = 2.0 | 148 UF = 2.0 |
| | 1024 | 1167 UF = 1.0 | 641 UF = 1.0 | 378 UF = 1.0 | 246 UF = 1.0 | 290 UF = 2.0 | 246 UF = 2.0 | 180 UF = 2.0 |
| | 2048 | 2191 UF = 1.0 | 1153 UF = 1.0 | 634 UF = 1.0 | 374 UF = 1.0 | 461 UF = 2.0 | 374 UF = 2.0 | 244 UF = 2.0 |
| | 4096 | 4239 UF = 1.0 | 2177 UF = 1.0 | 1146 UF = 1.0 | 630 UF = 1.0 | 802 UF = 2.0 | 630 UF = 2.0 | 372 UF = 2.0 |

H.2 Ack Latency §

Ack Latency tables were simplified in the 4.0 Revision of this specification. At that time, the original tables were moved to this appendix. Note that in the 4.0 Revision of this specification, Tables 3-9 and 3-10 were distinct, but identical tables. The 5.0 Revision contains a single table that applies to 8.0 GT/s or higher.

| Original Tables | | Simplified Tables | |
|-----------------|-------------------|-------------------|-------------------|
| Base 3.1 | Base 4.0 or later | Base 4.0 | Base 5.0 or later |
| Table 3-7 | § Table H-5 | Table 3-7 | § Table 3-10 |
| Table 3-8 | § Table H-6 | Table 3-8 | § Table 3-11 |
| Table 3-9 | § Table H-7 | Table 3-9 | § Table 3-12 |
| n/a | n/a | Table 3-10 | |

The Maximum Ack Latency limits are calculated using the formula:

$$\frac{(\text{Rx_MPS_Limit} + \text{TLPOverhead}) \times \text{UpdateFactor}}{\text{LinkWidth}} + \text{InternalDelay}$$

Equation H-2 Max Ack Latency §

where:

Rx_MPS_Limit

The Rx_MPS_Limit value as determined in § Section 2.2.2 . For a Multi-Function Device where different Functions have different Rx_MPS_Limit values, using the smallest Rx_MPS_Limit value across all Functions is strongly recommended.

TLPOverhead

represents the additional TLP components which consume Link bandwidth (TLP Prefix, header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.

AckFactor

represents the number of maximum size TLPs which can be received before an Ack is sent, and is used to balance Link bandwidth efficiency and retry buffer size - the value varies according to Rx_MPS_Limit and Link width. Table H-4, Table H-5, and Table H-6 below include the AckFactor(AF).

LinkWidth

is the operating width of the Link.

InternalDelay

represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times for 2.5 GT/s mode operation, 70 Symbol Times for 5.0 GT/s operation, and 115 Symbol Times for 8.0 GT/s and 16.0 GT/s operation.

Table H-5 Maximum Ack Latency Limit and AckFactor for 2.5 GT/s (Symbol Times) §

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

Base 6.4 vs Base 6.3

Table H-6 Maximum Ack Transmission Latency Limit and AckFactor for 5.0 GT/s (Symbol Times) §

| | | Link Operating Width | | | | | | |
|-------------------------|------|----------------------|------------------|------------------|-----------------|-----------------|-----------------|-----------------|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Rx_MPS_Limit (bytes) | 128 | 288 AF = 1.4 | 179 AF = 1.4 | 124 AF = 1.4 | 118 AF = 2.5 | 109 AF = 3.0 | 99 AF = 3.0 | 84 AF = 3.0 |
| | 256 | 467 AF = 1.4 | 268 AF = 1.4 | 169 AF = 1.4 | 158 AF = 2.5 | 141 AF = 3.0 | 123 AF = 3.0 | 96 AF = 3.0 |
| | 512 | 610 AF = 1.0 | 340 AF = 1.0 | 205 AF = 1.0 | 137 AF = 1.0 | 160 AF = 2.0 | 137 AF = 2.0 | 103 AF = 2.0 |
| | 1024 | 1122 AF = 1.0 | 596 AF = 1.0 | 333 AF = 1.0 | 201 AF = 1.0 | 245 AF = 2.0 | 201 AF = 2.0 | 135 AF = 2.0 |
| | 2048 | 2146 AF = 1.0 | 1108 AF = 1.0 | 589 AF = 1.0 | 329 AF = 1.0 | 416 AF = 2.0 | 329 AF = 2.0 | 199 AF = 2.0 |
| | 4096 | 4194 AF = 1.0 | 2132 AF = 1.0 | 1101 AF = 1.0 | 585 AF = 1.0 | 757 AF = 2.0 | 585 AF = 2.0 | 327 AF = 2.0 |

Table H-7 Maximum Ack Transmission Latency Limit and AckFactor for 8.0 GT/s (Symbol Times) §

| | | Link Operating Width | | | | | | |
|-------------------------|------|----------------------|------------------|------------------|-----------------|-----------------|-----------------|-----------------|
| | | x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Rx_MPS_Limit (bytes) | 128 | 333 AF = 1.4 | 224 AF = 1.4 | 169 AF = 1.4 | 163 AF = 2.5 | 154 AF = 3.0 | 144 AF = 3.0 | 129 AF = 3.0 |
| | 256 | 512 AF = 1.4 | 313 AF = 1.4 | 214 AF = 1.4 | 203 AF = 2.5 | 186 AF = 3.0 | 168 AF = 3.0 | 141 AF = 3.0 |
| | 512 | 655 AF = 1.0 | 385 AF = 1.0 | 250 AF = 1.0 | 182 AF = 1.0 | 205 AF = 2.0 | 182 AF = 2.0 | 148 AF = 2.0 |
| | 1024 | 1167 AF = 1.0 | 641 AF = 1.0 | 378 AF = 1.0 | 246 AF = 1.0 | 290 AF = 2.0 | 246 AF = 2.0 | 180 AF = 2.0 |
| | 2048 | 2191 AF = 1.0 | 1153 AF = 1.0 | 634 AF = 1.0 | 374 AF = 1.0 | 461 AF = 2.0 | 374 AF = 2.0 | 244 AF = 2.0 |
| | 4096 | 4239 AF = 1.0 | 2177 AF = 1.0 | 1146 AF = 1.0 | 630 AF = 1.0 | 802 AF = 2.0 | 630 AF = 2.0 | 372 AF = 2.0 |

Base 6.4 vs Base 6.3

I. Async Hot-Plug Reference Model §

This appendix presents a recommended reference model for async hot-plug. The reference model covers three areas:

- Async hot-plug initial configuration
- Async removal configuration and interrupt handling
- Async hot-add configuration and interrupt handling

There are no normative requirements in this section. The entire reference model is contained in a series of implementation notes. The reference model documents how the various hot-plug mechanisms are envisioned be used to implement a robust asynchronous hot-plug model, but does not mandate that they be used that way.

For brevity and readability, this section uses the following acronyms:

DLL

Data Link Layer

DSP

Downstream Port

FWF

firmware first

HPS

Hot-Plug Surprise

OOB

out-of-band

OS

operating system

PD

presence detect

SFW

system firmware

The reference model covers both the HPS and DPC mechanisms. DPC is the recommended mechanism. The reference model covers FWF for DPC but not for HPS.

The reference model provides a recommended framework for DPC software to support Containment Error Recovery (CER) along with async hot-plug. The reference model does not explicitly cover CER outside of async hot-plug, but certain aspects can be leveraged for DPC support of CER when async hot-plug is not being used.

SFW may use the System Firmware Intermediary (SFI) Capability for async hot-plug, orderly removal hot-plug, or other operations. This reference model does not rely on its functionality nor cover its use.

The reference model refers to various bits or fields outlined in § Section 6.7.2. For brevity, pointers to the associated registers are not replicated in this section.

IMPLEMENTATION NOTE: IN-BAND PRESENCE DETECT MECHANISM DEPRECATED FOR ASYNC HOT-PLUG §

Due to architectural issues, the in-band (Physical-Layer-based) portion of the PD mechanism is deprecated for use with async hot-plug. One issue is that in-band PD as architected does not detect adapter removal during certain LTSSM states, notably the L1 and Disabled States. Another issue is that when both in-band and OOB PD are being used together, the Presence Detect State bit and its associated interrupt mechanism always reflect the logical OR of the inband and OOB PD states, and with some hot-plug hardware configurations, it is important for software to detect and respond to in-band and OOB PD events independently.

If OOB PD is being used and the associated DSP supports In-Band PD Disable, it is recommended that the In-Band PD Disable bit be Set, and the Presence Detect State bit and its associated interrupt mechanism be used exclusively for OOB PD.

As a substitute for in-band PD with async hot-plug, the reference model uses either the DPC or the DLL Link Active mechanism.

The reference model assumes and covers the configurations listed below. While these cover the bulk of the envisioned use cases, many minor variations are not explicitly covered. For example, there are multiple ways to determine if a slot supports OOB PD, and the reference model does not cover them. As another example, the reference model refers to adding or removing an adapter, but some hot-pluggable modules may include a Switch and multiple Endpoint components.

- Reference model assumptions:
 - DSPs support DLL Link Active Reporting
 - DSPs support In-Band PD Disable
 - Operating systems support both HPS and DPC, using DPC if available
- Reference model covers:
 - Slots that support HPS and/or DPC (SW never enables both at same time)
 - Slots with or without OOB PD
 - RPs with or without RP Extensions for DPC

I.1 Async Hot-Plug Initial Configuration §

IMPLEMENTATION NOTE: ASYNC HOT-PLUG INITIAL CONFIGURATION §

| Basic Steps | HPS | DPC |
|--|--|-----|
| Determine capability control entity | <ul style="list-style-type: none"> OS requests, and is granted control of PCIe Native Hot-Plug If FWF, SFW retains control of AER and DPC Else OS requests and is granted control of AER and DPC | |
| OS and SFW determine which async hot-plug mechanism to use; OS/SFW interactions here are outside scope of this specification | <ul style="list-style-type: none"> If DPC capability then <ul style="list-style-type: none"> If HPS bit not Set, use DPC Else attempt to Clear HPS bit (§ Section 6.7.4.4) <ul style="list-style-type: none"> If successful, use DPC Else use HPS Else if HPS bit Set, use HPS Else async hot-plug cannot be supported by this slot Configure DSP for selected mechanism | |
| OS determines if adapter present | <ul style="list-style-type: none"> If OOB PD supported, use it to determine if adapter is present <ul style="list-style-type: none"> Set In-band PD Disable bit (SFW may have it Set by default) Read PD State bit Else allow Link to attempt to train; use DLL Link Active to determine if adapter is present (and at least minimally functional) | |
| If adapter is present, OS waits for adapter to become ready for configuration | <ul style="list-style-type: none"> If using Device Readiness Status (DRS), begin configuration if/after DSP receives a DRS Message If using Configuration RRS Software Visibility, can attempt first Configuration Read (of Vendor ID field) after 100 ms Otherwise, must wait at least 1000 ms before attempting first Configuration Read | |
| OS configures for appropriate case | <ul style="list-style-type: none"> If adapter is present, configure system for handling async removal Else configure system for handling async hot-add | |

I.2 Async Removal Configuration and Interrupt Handling §

IMPLEMENTATION NOTE: ASYNC REMOVAL CONFIGURATION §

| Basic Steps | HPS | DPC |
|---|--|---|
| OS/SFW configure appropriate error handling | <ul style="list-style-type: none"> If OS and driver support a non-CER error recovery approach, its policies may influence some of these error settings Configure the error handling supported by HPS | <ul style="list-style-type: none"> If OS and driver support CER, its policies may influence some of these error settings Enable uncorrectable AER errors to trigger DPC, including <u>Surprise Down</u> If using RP Extensions for DPC, configure RP PIO error handling Configure RP Completion Timeout handling per platform and OS policies |
| OS/SFW configure async removal interrupts | <ul style="list-style-type: none"> OS enables DLL State Changed interrupt | <ul style="list-style-type: none"> If FWF, configure DPC for ERR_COR signaling to enable SFW handling Else configure DPC for interrupt to enable OS handling |

IMPLEMENTATION NOTE: ASYNC REMOVAL INTERRUPT HANDLING §

| Basic Steps | HPS | DPC |
|---|---|---|
| Service routine entry | <ul style="list-style-type: none"> If PD Changed or DLL State Change, OS is interrupted | <ul style="list-style-type: none"> If PD Changed, OS is interrupted <ul style="list-style-type: none"> If FWF, OS invokes SFW; preferably via OS Setting DPC Software Trigger bit (if implemented) If DPC triggered <ul style="list-style-type: none"> If FWF, DPC sends <u>ERR_COR</u> to signal SFW Else DPC interrupts OS |
| Prevent further Link activity | <ul style="list-style-type: none"> OS Sets Disable Link; Link goes down and stays down OS polls DLL Link Active until Clear | <ul style="list-style-type: none"> DPC automatically keeps Link down |
| Log errors from DSP's AER/DPC | | <ul style="list-style-type: none"> If FWF, SFW logs DSP errors <ul style="list-style-type: none"> SFW invokes OS and exits OS logs and then clears DSP errors |
| OS notifies impacted software/driver, which cease accessing the adapter | | |
| OS determines if adapter is still present | | <ul style="list-style-type: none"> Some FFs with OOB PD automatically power off slot and/or disable switched signals If OOB PD supported, use it to determine if adapter is physically present If adapter not determined to be absent, re-enable the Link and slot <ul style="list-style-type: none"> As applicable, Clear Disable Link or release DPC Wait until Link trains or adapter is deemed absent or non-functional If non-functional, optionally perform a slot reset (if supported) If still non-functional, optionally power-cycle the slot (if supported) |
| If OS determined adapter to be present | | <ul style="list-style-type: none"> OS waits for adapter to become ready for configuration OS enumerates and configures the adapter If DPC and FWF, OS invokes SFW to log adapter AER errors |

| Basic Steps | HPS | DPC |
|----------------------------|--|-----|
| | <ul style="list-style-type: none"> • OS logs adapter AER errors and then clears them • If OS determines the adapter is suitable for continued operation <ul style="list-style-type: none"> ◦ OS configures for async removal handling ◦ OS resumes driver • Else OS takes OS-specific action | |
| Else (adapter not present) | <ul style="list-style-type: none"> • OS ensures DSP is in a clean state ready for a new/different adapter • OS configures for async hot-add handling | |

Base 6.4 vs Base 6.3

I.3 Async Hot-Add Configuration and Interrupt Handling §

IMPLEMENTATION NOTE: ASYNC HOT-ADD CONFIGURATION §

| Basic Steps | HPS | DPC |
|---|-----|-----|
| OS configures for async hot-add handling | | |
| <ul style="list-style-type: none"> • Enable Link to train if/when an adapter is inserted <ul style="list-style-type: none"> ◦ E.g., Clear Disable Link or release DPC if needed • If appropriate for form factor, enable power controller prior to adapter insertion • If OOB PD supported, enable OS interrupt on PD Changed • Else enable OS interrupt on DLL State Changed | | |

IMPLEMENTATION NOTE: ASYNC HOT-ADD INTERRUPT HANDLING §

| Basic Steps | HPS | DPC |
|--|-----|-----|
| OS is interrupted due to PD Changed or DLL State Changed | | |
| OS waits for the adapter to become ready for configuration <ul style="list-style-type: none">• If appropriate for form factor, enable power controller now (following adapter insertion)• Wait for DLL Link Active• Wait for adapter to become ready for configuration | | |
| OS configures adapter per standard OS conventions | | |
| OS configures for async removal handling | | |
| OS calls driver to complete adapter configuration and begin normal operation | | |

Base 6.4 vs Base 6.3

J. Alpha Power and Reverse lookup assignment §

The following files are attached to the No Changebar PDF for this specification.

- [alpha_powers.vh](#)
 - MD5 = f25de382b6f00fbcefa00a4d25199e85
- [ecc_84to86_encoder.sv](#)
 - MD5 = 38956a4b15be17181b2cb098c000ff85
- [ecc_250to256_encoder.sv](#)
 - MD5 = 55e41b4727c51715d60b8a08c64d25bb
- [ecc_86to84_decoder.sv](#)
 - MD5 = 35518f84edf9361688f065394ceff0cb
- [ecc_256to250_decoder.sv](#)
 - MD5 = dbdf0bda8eef74cb9d7af00312a7ea53

Base 6.4 vs Base 6.3

J.1 Alpha Powers §

alpha_powers.vh

Base 6.4 vs Base 6.3

```
//index to alpha power lookup
localparam logic [7:0] i_to_alpha_power_i[255:0] =
'{
8'h01,
8'h8e,
8'h47,
8'had,
8'hd8,
8'h6c,
8'h36,
8'h1b,
8'h83,
8'hcf,
8'he9,
8'hfa,
8'h7d,
8'hb0,
8'h58,
8'h2c,
8'h16,
8'h0b,
8'h8b,
8'hcb,
8'heb,
8'hfb,
8'hf3,
8'hf7,
8'hf5,
8'hf4,
8'h7a,
8'h3d,
8'h90,
8'h48,
8'h24,
8'h12,
8'h09,
8'h8a,
8'h45,
8'hac,
8'h56,
8'h2b,
8'h9b,
8'hc3,
8'hef,
8'hf9,
8'hf2,
8'h79,
8'hb2,
8'h59,
8'ha2,
8'h51,
8'ha6,
8'h53,
8'ha7,
8'hdd,
8'he0,
8'h70,
8'h38,
8'h1c,
8'h0e,
8'h07,
8'h8d,
8'hc8,
8'h64,
```

Base 6.4 vs Base 6.3

8'h32,
8'h19,
8'h82,
8'h41,
8'hae,
8'h57,
8'ha5,
8'hdc,
8'h6e,
8'h37,
8'h95,
8'hc4,
8'h62,
8'h31,
8'h96,
8'h4b,
8'hab,
8'hdb,
8'he3,
8'hff,
8'hf1,
8'hf6,
8'h7b,
8'hb3,
8'hd7,
8'he5,
8'hfc,
8'h7e,
8'h3f,
8'h91,
8'hc6,
8'h63,
8'hbf,
8'hd1,
8'he6,
8'h73,
8'hb7,
8'hd5,
8'he4,
8'h72,
8'h39,
8'h92,
8'h49,
8'haa,
8'h55,
8'ha4,
8'h52,
8'h29,
8'h9a,
8'h4d,
8'ha8,
8'h54,
8'h2a,
8'h15,
8'h84,
8'h42,
8'h21,
8'h9e,
8'h4f,
8'ha9,
8'hda,
8'h6d,
8'hb8,
8'h5c,
8'h2e,

Base 6.4 vs Base 6.3

8'h17,
8'h85,
8'hcc,
8'h66,
8'h33,
8'h97,
8'hc5,
8'hec,
8'h76,
8'h3b,
8'h93,
8'hc7,
8'hed,
8'hf8,
8'h7c,
8'h3e,
8'h1f,
8'h81,
8'hce,
8'h67,
8'hbd,
8'hd0,
8'h68,
8'h34,
8'h1a,
8'h0d,
8'h88,
8'h44,
8'h22,
8'h11,
8'h86,
8'h43,
8'haf,
8'hd9,
8'he2,
8'h71,
8'hb6,
8'h5b,
8'ha3,
8'hdf,
8'he1,
8'hfe,
8'h7f,
8'hb1,
8'hd6,
8'h6b,
8'hbb,
8'hd3,
8'he7,
8'hfd,
8'hf0,
8'h78,
8'h3c,
8'h1e,
8'h0f,
8'h89,
8'hca,
8'h65,
8'hbc,
8'h5e,
8'h2f,
8'h99,
8'hc2,
8'h61,
8'hbe,

Base 6.4 vs Base 6.3

```
8'h5f,  
8'ha1,  
8'hde,  
8'h6f,  
8'hb9,  
8'hd2,  
8'h69,  
8'hba,  
8'h5d,  
8'ha0,  
8'h50,  
8'h28,  
8'h14,  
8'h0a,  
8'h05,  
8'h8c,  
8'h46,  
8'h23,  
8'h9f,  
8'hc1,  
8'hee,  
8'h77,  
8'hb5,  
8'hd4,  
8'h6a,  
8'h35,  
8'h94,  
8'h4a,  
8'h25,  
8'h9c,  
8'h4e,  
8'h27,  
8'h9d,  
8'hc0,  
8'h60,  
8'h30,  
8'h18,  
8'h0c,  
8'h06,  
8'h03,  
8'h8f,  
8'hc9,  
8'hea,  
8'h75,  
8'hb4,  
8'h5a,  
8'h2d,  
8'h98,  
8'h4c,  
8'h26,  
8'h13,  
8'h87,  
8'hcd,  
8'he8,  
8'h74,  
8'h3a,  
8'h1d,  
8'h80,  
8'h40,  
8'h20,  
8'h10,  
8'h08,  
8'h04,  
8'h02,  
8'h01
```

Base 6.4 vs Base 6.3

```
};

//Reverse index lookup
localparam logic [7:0] alpha_power_i_to_i[255:0] =
'{
8'haf,
8'h58,
8'h50,
8'ha8,
8'hea,
8'hf4,
8'hd6,
8'h74,
8'he8,
8'had,
8'he7,
8'he6,
8'he9,
8'hd5,
8'hae,
8'h4f,
8'hd7,
8'h2c,
8'h75,
8'h7a,
8'heb,
8'h16,
8'hf5,
8'h0b,
8'h51,
8'ha0,
8'ha9,
8'h9c,
8'hb0,
8'h5f,
8'h59,
8'hcb,
8'h5a,
8'h3e,
8'hcc,
8'hbb,
8'hb1,
8'h86,
8'h60,
8'hfb,
8'haa,
8'h55,
8'h9d,
8'h29,
8'h52,
8'h3b,
8'ha1,
8'h6c,
8'hf6,
8'h6f,
8'h0c,
8'h7f,
8'hec,
8'h49,
8'h17,
8'hc4,
8'h76,
8'ha4,
8'h7b,
8'hb7,
8'hd8,
```

Base 6.4 vs Base 6.3

8'h43,
8'h2d,
8'h1f,
8'ha2,
8'h41,
8'h6d,
8'h47,
8'h53,
8'h39,
8'h3c,
8'h84,
8'h9e,
8'h5d,
8'h2a,
8'h14,
8'hab,
8'hd3,
8'h56,
8'hf2,
8'h61,
8'hbe,
8'hfc,
8'hdc,
8'hb2,
8'h97,
8'h87,
8'h90,
8'hcd,
8'hcf,
8'hbc,
8'h95,
8'h5b,
8'hd1,
8'h3f,
8'h37,
8'h2e,
8'h89,
8'h20,
8'h23,
8'hd9,
8'h92,
8'h44,
8'h11,
8'h7c,
8'hb4,
8'hb8,
8'h26,
8'h77,
8'h99,
8'ha5,
8'he3,
8'h18,
8'hfe,
8'hc5,
8'h31,
8'hed,
8'hde,
8'h4a,
8'h67,
8'h0d,
8'h63,
8'h80,
8'h8c,
8'hf7,
8'hc0,

Base 6.4 vs Base 6.3

8'h70,
8'h07,
8'h57,
8'ha7,
8'hf3,
8'h73,
8'hac,
8'he5,
8'hd4,
8'h4e,
8'h2b,
8'h79,
8'h15,
8'h0a,
8'h9f,
8'h9b,
8'h5e,
8'hca,
8'h3d,
8'hba,
8'h85,
8'hfa,
8'h54,
8'h28,
8'h3a,
8'h6b,
8'h6e,
8'h7e,
8'h48,
8'hc3,
8'ha3,
8'hb6,
8'h42,
8'h1e,
8'h40,
8'h46,
8'h38,
8'h83,
8'h5c,
8'h13,
8'hd2,
8'hf1,
8'hbd,
8'hdb,
8'h96,
8'h8f,
8'hce,
8'h94,
8'hd0,
8'h36,
8'h88,
8'h22,
8'h91,
8'h10,
8'hb3,
8'h25,
8'h98,
8'he2,
8'hfd,
8'h30,
8'hdd,
8'h66,
8'h62,
8'h8b,
8'hbf,

Base 6.4 vs Base 6.3

8'h06,
8'ha6,
8'h72,
8'he4,
8'h4d,
8'h78,
8'h09,
8'h9a,
8'hc9,
8'hb9,
8'hf9,
8'h27,
8'h6a,
8'h7d,
8'hc2,
8'hb5,
8'h1d,
8'h45,
8'h82,
8'h12,
8'hf0,
8'hda,
8'h8e,
8'h93,
8'h35,
8'h21,
8'h0f,
8'h24,
8'he1,
8'h2f,
8'h65,
8'h8a,
8'h05,
8'h71,
8'h4c,
8'h08,
8'hc8,
8'hf8,
8'h69,
8'hc1,
8'h1c,
8'h81,
8'hef,
8'h8d,
8'h34,
8'h0e,
8'he0,
8'h64,
8'h04,
8'h4b,
8'hc7,
8'h68,
8'h1b,
8'hee,
8'h33,
8'hdf,
8'h03,
8'hc6,
8'h1a,
8'h32,
8'h02,
8'h19,
8'h01,
8'h00,

```
8'hff
};
```

J.2 84 Byte to 86 Byte Encoder §

ecc_84to86_encoder.sv

```
'ifndef ECC_84T086_ENCODER_SV
`define ECC_84T086_ENCODER_SV
//Description: This module implements the 84B to 86B ECC group encoder
//as defined in Chapter 4 of the PCI Express Base Specification 6.0.
//Input: 84 bytes of Data
//Output: 86 bytes of data. Check is on byte 84, Parity is on byte 85.
module ecc_84to86_encoder
(
    input logic [7:0] data_in[83:0],
    output logic [7:0] data_out[85:0]
);
logic [7:0] synd_parity;
logic [7:0] synd_check;
`include "alpha_powers.vh"
//Calculate the Check and Parity Bytes.
always_comb begin
    synd_parity      = 8'h00;
    synd_check       = 8'h00;
    for(int i = 0 ; i < 84 ; i++) begin
        synd_parity ^= data_in[i];
        synd_check   ^= ({8{data_in[i][0]}} & i_to_alpha_power_i[84-i]) ^
                        ({8{data_in[i][1]}} & i_to_alpha_power_i[85-i]) ^
                        ({8{data_in[i][2]}} & i_to_alpha_power_i[86-i]) ^
                        ({8{data_in[i][3]}} & i_to_alpha_power_i[87-i]) ^
                        ({8{data_in[i][4]}} & i_to_alpha_power_i[88-i]) ^
                        ({8{data_in[i][5]}} & i_to_alpha_power_i[89-i]) ^
                        ({8{data_in[i][6]}} & i_to_alpha_power_i[90-i]) ^
                        ({8{data_in[i][7]}} & i_to_alpha_power_i[91-i]));
    end
end
//Assign Output
always_comb begin
    for (int i = 0 ; i < 84 ; i++) begin
        data_out[i] = data_in[i];
    end
    data_out[84]    = synd_check;
    data_out[85]    = synd_parity;
end
endmodule
`endif
```

J.3 250 Byte to 256 Byte Encoder example §

ecc_250to256_encoder.sv

Base 6.4 vs Base 6.3

```

`ifndef ECC_250T0256_ENCODER__SV
`define ECC_250T0256_ENCODER__SV
//Description: This module implements ECC for Flit level blocks.
//It splits the input 250 bytes into the 3 groups defined by PCIe 6.0
//and assigns the ECC bytes based on the byte positions shown
//in Chapter 4 of PCIe 6.0 Specification.
//
//Input      : 250 bytes of Flit without ECC.
//Output     : 256 bytes of Flit with ECC added.
//Child Module : ecc_84to86_encoder
module ecc_250to256_encoder
(
    input logic [7:0] data_in[249:0],
    output logic [7:0] data_out[255:0]
);
logic [7:0] ecc_output0[85:0];
logic [7:0] ecc_output1[85:0];
logic [7:0] ecc_output2[85:0];
logic [7:0] ecc0_input[83:0];
logic [7:0] ecc1_input[83:0];
logic [7:0] ecc2_input[83:0];
//Split input into the 3 ECC groups
//Groups 1 and 2 get an extra byte of 0 padded to byte 83.
always_comb begin
    for (int i = 0 ; i < 83 ; i++) begin
        ecc0_input[i] = data_in[i*3];
        ecc1_input[i] = data_in[(i*3) + 1];
        ecc2_input[i] = data_in[(i*3) + 2];
    end
    ecc0_input[83]    = data_in[249];
    ecc1_input[83]    = 8'h00;
    ecc2_input[83]    = 8'h00;
end
ecc_84to86_encoder enc0 (
    .data_in ( ecc0_input ),
    .data_out ( ecc_output0 )
);
ecc_84to86_encoder enc1 (
    .data_in ( ecc1_input ),
    .data_out ( ecc_output1 )
);
ecc_84to86_encoder enc2 (
    .data_in ( ecc2_input ),
    .data_out ( ecc_output2 )
);
//Assign output
always_comb begin
    //First 250 bytes are assigned directly from input.
    for (int i = 0 ; i < 250 ; i++) begin
        data_out[i] = data_in[i];
    end
    //Next Six bytes are assigned using the check and parity
    //from the three ECC groups.
    data_out[250]    = ecc_output1[84]; //ECC1[0] = Check of Group 1.
    data_out[251]    = ecc_output2[84]; //ECC2[0] = Check of Group 2.
    data_out[252]    = ecc_output0[84]; //ECC0[0] = Check of Group 0.
    data_out[253]    = ecc_output1[85]; //ECC1[1] = Parity of Group 1.
    data_out[254]    = ecc_output2[85]; //ECC2[1] = Parity of Group 2.
    data_out[255]    = ecc_output0[85]; //ECC0[1] = Parity of Group 0.
end
endmodule
`endif

```

Base 6.4 vs Base 6.3

J.4 86 Byte to 84 Byte Decoder §

Base 6.4 vs Base 6.3

`ecc_86to84_decoder.sv`

```

`ifndef ECC_86T084_DECODER__SV
`define ECC_86T084_DECODER__SV //Description: This module implements the 86 byte
to 84 byte decoder for the //ECC code defined in PCIe 6.0 Specification. //Input : 86 bytes of encoded data //Outputs : // 84 bytes
of decoded data (corrected if applicable) // synd_check and synd parity for logging // Error magnitude - value of the error term //
Error byte - location of the error byte, not applicable if // (unc_error == 1) // no_error - no error was detected. // single_error -
single error was detected. // unc_error - uncorrectable error (error decoding pointed to // non-existent column). module
ecc_86to84_decoder ( input logic [7:0] data_in[85:0], output logic [7:0] data_out[83:0], output logic [7:0] synd_check, output logic
[7:0] synd_parity, output logic [7:0] error_magnitude, output logic [6:0] error_byte, output logic no_error, output logic
single_error, output logic unc_error ); logic synd_parity_0; logic synd_check_0; logic [7:0] synd_parity_map_i; logic [7:0]
synd_check_map_i; logic [8:0] error_byte_result; logic overflow; include "alpha_powers.vh" logic [7:0]
encoded_data[85:0]; ecc_84to86_encoder encoder (.data_in ( data_in[83:0] ), .data_out (
encoded_data[85:0] ) ); // The decoder has to first invoke the encoder function to calculate the
syndrome assign synd_parity = encoded_data[85] ^ data_in[85]; assign synd_check = encoded_data[84]
^ data_in[84]; assign synd_parity_0 = (synd_parity == 8'h00); assign synd_check_0 = (synd_check ==
8'h00); assign synd_parity_map_i = alpha_power_i_to_i[synd_parity]; assign synd_check_map_i =
alpha_power_i_to_i[synd_check]; assign error_byte = error_byte_result[6:0]; always_comb begin
no_error = 1'b0; single_error = 1'b0; unc_error = 1'b0; error_byte_result = 9'd0; error_magnitude =
synd_parity; overflow = 1'b0; for ( int i = 0 ; i < 84 ; i++ ) begin data_out[i] = data_in[i]; end
unique casez ({synd_check_0, synd_parity_0}) 2'b11: begin no_error = 1'b1; end 2'b10: begin
single_error = 1'b1; error_byte_result = 9'd85; end 2'b01: begin single_error = 1'b1;
error_magnitude = synd_check; error_byte_result = 9'd84; end 2'b00: begin //Overflow term in the
equations below is ignored, as those cases are //outside the correction capability of the ECC, and
//we rely on CRC to detect those. if (synd_check_map_i < synd_parity_map_i) begin {overflow,
error_byte_result} = 9'd84 - (9'd255 + {1'b0, synd_check_map_i} - {1'b0,synd_parity_map_i}); end
else begin {overflow, error_byte_result} = 9'd84 - ({1'b0,synd_check_map_i}
{1'b0,synd_parity_map_i} ); end if (error_byte_result >= 9'd84) begin unc_error = 1'b1; end else
begin single_error = 1'b1; data_out[error_byte_result] ^= synd_parity; end end endcase end
endmodule
`endif

```

Base 6.4 vs Base 6.3

```

`ifndef ECC_86T084_DECODER__SV
`define ECC_86T084_DECODER__SV
//Description: This module implements the 86 byte to 84 byte decoder for the
//ECC code defined in PCIe 6.0 Specification.
//
//Input   : 86 bytes of encoded data
//Outputs :
//      84 bytes of decoded data (corrected if applicable)
//      synd_check and synd parity for logging
//      Error magnitude - value of the error term
//      Error byte - location of the error byte, not applicable if
//                  (unc_error == 1)
//      no_error - no error was detected.
//      single_error - single error was detected.
//      unc_error - uncorrectable error (error decoding pointed to
//                  non-existent column).
module ecc_86to84_decoder
(
    input logic [7:0] data_in[85:0],
    output logic [7:0] data_out[83:0],
    output logic [7:0] synd_check,
    output logic [7:0] synd_parity,
    output logic [7:0] error_magnitude,
    output logic [6:0] error_byte,
    output logic no_error,
    output logic single_error,
    output logic unc_error
);
logic synd_parity_0;
logic synd_check_0;
logic [7:0] synd_parity_map_i;
logic [7:0] synd_check_map_i;
logic [8:0] error_byte_result;
logic overflow;
`include "alpha_powers.vh"
logic [7:0] encoded_data[85:0];
ecc_84to86_encoder encoder (
    .data_in ( data_in[83:0] ),
    .data_out ( encoded_data[85:0] )
); // The decoder has to first invoke the encoder function to calculate the syndrome
assign synd_parity      = encoded_data[85] ^ data_in[85];
assign synd_check       = encoded_data[84] ^ data_in[84];
assign synd_parity_0    = (synd_parity == 8'h00);
assign synd_check_0     = (synd_check == 8'h00);
assign synd_parity_map_i = alpha_power_i_to_i[synd_parity];
assign synd_check_map_i = alpha_power_i_to_i[synd_check];
assign error_byte       = error_byte_result[6:0];
always_comb begin
    no_error          = 1'b0;
    single_error      = 1'b0;
    unc_error         = 1'b0;
    error_byte_result = 9'd0;
    error_magnitude   = synd_parity;
    overflow          = 1'b0;
    for (int i = 0 ; i < 84 ; i++) begin
        data_out[i] = data_in[i];
    end
    unique casez ({synd_check_0, synd_parity_0})
    2'b11: begin
        no_error = 1'b1;
    end
    2'b10: begin
        single_error      = 1'b1;
        error_byte_result = 9'd85;
    end
end

```

Base 6.4 vs Base 6.3

```
end
2'b01: begin
    single_error      = 1'b1;
    error_magnitude   = synd_check;
    error_byte_result = 9'd84;
end
2'b00: begin
    //Overflow term in the equations below is ignored, as those cases are
    //outside the correction capability of the ECC, and
    //we rely on CRC to detect those.
    if (synd_check_map_i < synd_parity_map_i) begin
        {overflow, error_byte_result} = 9'd84 - (9'd255 + {1'b0, synd_check_map_i} - {1'b0,synd_parity_map_i});
    end else begin
        {overflow, error_byte_result} = 9'd84 - ({1'b0,synd_check_map_i} - {1'b0,synd_parity_map_i}) ;
    end
    if (error_byte_result >= 9'd84) begin
        unc_error          = 1'b1;
    end else begin
        single_error        = 1'b1;
        data_out[error_byte_result] ^= synd_parity;
    end
end
endcase
end
endmodule
`endif
```

Base 6.4 vs Base 6.3

J.5 256 Byte to 250 Byte decoder §

Base 6.4 vs Base 6.3

ecc_256to250_decoder.sv

```

`ifndef ECC_256TO250_DECODER_SV
`define ECC_256TO250_DECODER_SV //Description : This module takes in 256 bytes
//of the flit, and outputs 250 bytes //of corrected data. It also gives the metrics of the detected error per ECC group //defined in PCIe
6.0 Specification. //Note: it is possible that the ECC code corrupted the data further if more than a //single byte error happened
within an ECC group. The final check is the 8B-CRC //data should only be consumed if CRC passes post FEC correction. //Input:
256 bytes of the flit. //Output : 250 bytes of corrected data // synd_check and parity for each ECC group for logging (see signal
descriptions) // Error metrics for each ECC group (see signal descriptions). module ecc_256to250_decoder ( input logic [7:0]
data_in[255:0], output logic [7:0] data_out[249:0], output logic [7:0] synd_check0, //Syndrome Check for ECC Group 0. Flit Error
log 1, bits 31:24. output logic [7:0] synd_parity0, //Syndrom Parity for ECC Group 0. Flit Error log 1, bits 23:16. output logic [7:0]
synd_check1, //Syndrome Check for ECC Group 1. Flit Error log 2, bits 15:8. output logic [7:0] synd_parity1, //Syndrom Parity for
ECC Group 1. Flit Error log 2, bits 7:0. output logic [7:0] synd_check2, //Syndrome Check for ECC Group 2. Flit Error log 2, bits 31:24.
output logic [7:0] synd_parity2, //Syndrom Parity for ECC Group 2. Flit Error log 2, bits 23:16. //Error metrics //Note that error
metrics can be incorrect if more than //1 byte per ECC group is corrupted, and CRC check ultimately //determines whether the flit
is consumed or replayed. //If unc_error0 == 0, error_byte0 indicates the error location for ECC Group 0 as the //byte position
between(including) 0 and 85. //If unc_error1 == 0, error_byte1 indicates the error location for ECC Group 1 as the //byte position
between(including) 0 and 85. //If unc_error2 == 0, error_byte2 indicates the error location for ECC Group 2 as the //byte position
between(including) 0 and 85. //ECC Group 0 metrics : output logic [7:0] error_magnitude0, output logic [6:0] error_byte0, output
logic no_error0, output logic single_error0, output logic unc_error0, //ECC Group 1 metrics : output logic [7:0] error_magnitude1,
output logic [6:0] error_byte1, output logic no_error1, output logic single_error1, output logic unc_error1, //ECC Group 2 metrics :
output logic [7:0] error_magnitude2, output logic [6:0] error_byte2, output logic no_error2, output logic single_error2, output
logic unc_error2 ); logic [7:0] dec_data_out0[83:0]; logic [7:0] dec_data_out1[83:0]; logic [7:0] dec_data_out2[83:0]; logic [7:0]
dec_data_in0[85:0]; logic [7:0] dec_data_in1[85:0]; logic [7:0] dec_data_in2[85:0]; //Split the received flit into the 3 ECC groups.
always_comb begin for (int i=0;i<83;i++) begin dec_data_in0[i] = data_in[i*3]; dec_data_in1[i] = data_in[(i*3)+1]; dec_data_in2[i] =
data_in[(i*3)+2]; data_out[i*3] = dec_data_out0[i]; data_out[i*3+1] = dec_data_out1[i]; data_out[i*3+2] = dec_data_out2[i]; end
dec_data_in0[83] = data_in[249]; //Blue ECC group 0 dec_data_in0[84] = data_in[252]; dec_data_in0[85] = data_in[255];
dec_data_in1[83] = 8'h00; //Orange ECC group 1 dec_data_in1[84] = data_in[250]; dec_data_in1[85] = data_in[253];
dec_data_in2[83] = 8'h00; //Green ECC group 2 dec_data_in2[84] = data_in[251]; dec_data_in2[85] = data_in[254]; data_out[249] =
dec_data_out0[83]; end //Instantiate separate 86to84 decoder for each group. ecc_86to84_decoder dec0 (.data_in ( dec_data_in0 ),
.data_out ( dec_data_out0 ), .synd_check ( synd_check0 ), .synd_parity ( synd_parity0 ), .error_magnitude ( error_magnitude0 ),
.error_byte ( error_byte0 ), .no_error ( no_error0 ), .single_error ( single_error0 ), .unc_error ( unc_error0 )); ecc_86to84_decoder
dec1 (.data_in ( dec_data_in1 ), .data_out ( dec_data_out1 ), .synd_check ( synd_check1 ), .synd_parity ( synd_parity1 ),
.error_magnitude ( error_magnitude1 ), .error_byte ( error_byte1 ), .no_error ( no_error1 ), .single_error ( single_error1 ),
.unc_error ( unc_error1 )); ecc_86to84_decoder dec2 (.data_in ( dec_data_in2 ), .data_out ( dec_data_out2 ), .synd_check (
synd_check2 ), .synd_parity ( synd_parity2 ), .error_magnitude ( error_magnitude2 ), .error_byte ( error_byte2 ), .no_error (
no_error2 ), .single_error ( single_error2 ), .unc_error ( unc_error2 )); endmodule `endif

```

Base 6.4 vs Base 6.3

```

`ifndef ECC_256T0250_DECODER__SV
`define ECC_256T0250_DECODER__SV
//Description : This module takes in 256 bytes of the flit, and outputs 250 bytes
//of corrected data. It also gives the metrics of the detected error per ECC group
//defined in PCIe 6.0 Specification.
//Note: it is possible that the ECC code corrupted the data further if more than a
//single byte error happened within an ECC group. The final check is the 8B CRC -
//data should only be consumed if CRC passes post FEC correction.
//Input : 256 bytes of the flit.
//Output : 250 bytes of corrected data
//           synd_check and parity for each ECC group for logging (see signal descriptions)
//           Error metrics for each ECC group (see signal descriptions).
module ecc_256to250_decoder (
    input logic [7:0] data_in[255:0],
    output logic [7:0] data_out[249:0],
    output logic [7:0] synd_check0, //Syndrome Check for ECC Group 0. Flit Error log 1, bits 31:24.
    output logic [7:0] synd_parity0,//Syndrome Parity for ECC Group 0. Flit Error log 1, bits 23:16.
    output logic [7:0] synd_check1, //Syndrome Check for ECC Group 1. Flit Error log 2, bits 15:8.
    output logic [7:0] synd_parity1,//Syndrome Parity for ECC Group 1. Flit Error log 2, bits 7:0.
    output logic [7:0] synd_check2, //Syndrome Check for ECC Group 2. Flit Error log 2, bits 31:24.
    output logic [7:0] synd_parity2,//Syndrome Parity for ECC Group 2. Flit Error log 2, bits 23:16.
    //Error metrics
    //Note that error metrics can be incorrect if more than
    //1 byte per ECC group is corrupted, and CRC check ultimately
    //determines whether the flit is consumed or replayed.
    //
    //If unc_error0 == 0, error_byte0 indicates the error location for ECC Group 0 as the
    //byte position between(including) 0 and 85.
    //If unc_error1 == 0, error_byte1 indicates the error location for ECC Group 1 as the
    //byte position between(including) 0 and 85.
    //If unc_error2 == 0, error_byte2 indicates the error location for ECC Group 2 as the
    //byte position between(including) 0 and 85.
    //
    //ECC Group 0 metrics :
    output logic [7:0] error_magnitude0,
    output logic [6:0] error_byte0,
    output logic no_error0,
    output logic single_error0,
    output logic unc_error0,
    //ECC Group 1 metrics :
    output logic [7:0] error_magnitude1,
    output logic [6:0] error_byte1,
    output logic no_error1,
    output logic single_error1,
    output logic unc_error1,
    //ECC Group 2 metrics :
    output logic [7:0] error_magnitude2,
    output logic [6:0] error_byte2,
    output logic no_error2,
    output logic single_error2,
    output logic unc_error2
);
logic [7:0] dec_data_out0 [83:0];
logic [7:0] dec_data_out1 [83:0];
logic [7:0] dec_data_out2 [83:0];
logic [7:0] dec_data_in0 [85:0];
logic [7:0] dec_data_in1 [85:0];
logic [7:0] dec_data_in2 [85:0];
//Split the received flit into the 3 ECC groups.
always_comb begin
    for (int i=0;i<83;i++) begin
        dec_data_in0[i] = data_in[i*3];
        dec_data_in1[i] = data_in[(i*3) + 1];
        dec_data_in2[i] = data_in[(i*3) + 2];
    end
end

```

Base 6.4 vs Base 6.3

```

    data_out[i*3]      = dec_data_out0[i];
    data_out[i*3 + 1] = dec_data_out1[i];
    data_out[i*3 + 2] = dec_data_out2[i];
end
dec_data_in0[83]      = data_in[249]; //Blue ECC group 0
dec_data_in0[84]      = data_in[252];
dec_data_in0[85]      = data_in[255];
dec_data_in1[83]      = 8'h00;           //Orange ECC group 1
dec_data_in1[84]      = data_in[250];
dec_data_in1[85]      = data_in[253];
dec_data_in2[83]      = 8'h00;           //Green ECC group 2
dec_data_in2[84]      = data_in[251];
dec_data_in2[85]      = data_in[254];
data_out[249]         = dec_data_out0[83];
end
//Instantiate separate 86to84 decoder for each group.
ecc_86to84_decoder dec0 (
    .data_in ( dec_data_in0 ),
    .data_out ( dec_data_out0 ),
    .synd_check ( synd_check0 ),
    .synd_parity ( synd_parity0 ),
    .error_magnitude ( error_magnitude0 ),
    .error_byte ( error_byte0 ),
    .no_error ( no_error0 ),
    .single_error ( single_error0 ),
    .unc_error ( unc_error0 )
);
ecc_86to84_decoder dec1 (
    .data_in ( dec_data_in1 ),
    .data_out ( dec_data_out1 ),
    .synd_check ( synd_check1 ),
    .synd_parity ( synd_parity1 ),
    .error_magnitude ( error_magnitude1 ),
    .error_byte ( error_byte1 ),
    .no_error ( no_error1 ),
    .single_error ( single_error1 ),
    .unc_error ( unc_error1 )
);
ecc_86to84_decoder dec2 (
    .data_in ( dec_data_in2 ),
    .data_out ( dec_data_out2 ),
    .synd_check ( synd_check2 ),
    .synd_parity ( synd_parity2 ),
    .error_magnitude ( error_magnitude2 ),
    .error_byte ( error_byte2 ),
    .no_error ( no_error2 ),
    .single_error ( single_error2 ),
    .unc_error ( unc_error2 )
);
endmodule
`endif

```

Base 6.4 vs Base 6.3

K. MATLAB created generator matrix for CRC code §

The following files are attached to the No Changebar PDF for this specification.

- [gen_matrix.txt](#)
 - MD5 = e6d5d08c0fc89cf28bcd142bcd9c635
- [gen_matrix.v](#)
 - MD5 = 44362bcba204ff1548c3fbdc61d8efdb
- [pci_sig_8B_crc.sv](#)
 - MD5 = 04f8a99644d87483155921ce870a09f1

K.1 Generator Matrix output §

`gen_matrix.txt` (8 columns and 1936 rows, values in hex):

Base 6.4 vs Base 6.3

```

a7 ad 46 a7 3e 67 9d 2d
c6 c3 23 c6 1f a6 db 83
63 f4 84 63 9a 53 f8 d4
a4 7a 42 a4 4d bc 7c 6a
52 3d 21 52 b3 5e 3e 35
29 8b 85 29 cc 2f 1f 8f
81 d0 d7 81 66 82 9a d2
d5 68 fe d5 33 41 4d 69
14 2c c9 87 5a 45 80 cd
0a 16 f1 d6 2d b7 40 f3
05 0b ed 6b 83 ce 20 ec
97 90 e3 a0 d4 67 10 76
de 48 e4 50 6a a6 08 3b
6f 24 72 28 35 53 04 88
a2 12 39 14 8f bc 02 44
51 09 89 0a d2 5e 01 22
a1 38 40 d7 c4 13 ae e5
c5 1c 20 fe 62 9c 57 e7
f7 0e 10 7f 31 4e be e6
ee 07 08 aa 8d 27 5f 73
77 96 04 55 d3 86 ba ac
ae 4b 02 bf fc 43 5d 56
57 b0 01 ca 7e b4 bb 2b
be 58 95 65 3f 5a c8 80
29 71 eb d5 84 db cd 90
81 ad e0 ff 42 f8 f3 48
d5 c3 70 ea 21 7c ec 24
ff f4 38 75 85 3e 76 12
ea 7a 1c af d7 1f 3b 09
75 3d 0e c2 fe 9a 88 91
af 8b 07 61 7f 4d 44 dd
c2 d0 96 a5 aa b3 22 fb
95 4a 8e 60 c1 81 1b 88
df 25 47 30 f5 d5 98 44
fa 87 b6 18 ef ff 4c 22
7d d6 5b 0c e2 ea 26 11
ab 6b b8 06 71 75 13 9d
c0 a0 5c 03 ad af 9c db
60 50 2e 94 c3 c2 4e f8
30 28 17 4a f4 61 27 7c
b5 ba 1f 3e 0d ae 3b a1
cf 5d 9a 1f 93 57 88 c5
f2 bb 4d 9a dc be 44 f7
79 c8 b3 4d 6e 5f 22 ee
a9 64 cc b3 37 ba 11 77
c1 32 66 cc 8e 5d 9d ae
f5 19 33 66 47 bb db 57
ef 99 8c 33 b6 c8 f8 be
26 df c5 91 b8 dd ea 75
13 fa f7 dd 5c fb 75 af
9c 7d ee fb 2e e8 af c2
4e ab 77 e8 17 74 c2 61
27 c0 ae 74 9e 3a 61 a5
86 60 57 3a 4f 1d a5 c7
43 30 be 1d b2 9b c7 f6
b4 18 5f 9b 59 d8 f6 7b
14 6a 47 73 ed 14 08 89
0a 35 b6 ac e3 0a 04 d1
05 8f 5b 56 e4 05 02 fd
97 d2 b8 2b 72 97 01 eb
de 69 5c 80 39 de 95 e0
6f a1 2e 40 89 6f df 70
a2 c5 17 20 d1 a2 fa 38
51 f7 9e 10 fd 51 7d 1c

```

Base 6.4 vs Base 6.3

```

e7 b6 b4 60 95 9b ea e5
e6 5b 5a 30 df d8 75 e7
73 b8 2d 18 fa 6c af e6
ac 5c 83 0c 7d 36 c2 73
56 2e d4 06 ab 1b 61 ac
2b 17 6a 03 c0 98 a5 56
80 9e 35 94 60 4c c7 2b
40 4f 8f 4a 30 26 f6 80
c9 1d 2d ea b9 94 93 4e
f1 9b 83 75 c9 4a dc 27
ed d8 d4 af f1 25 6e 86
e3 6c 6a c2 ed 87 37 43
e4 36 35 61 e3 d6 8e b4
72 1b 8f a5 e4 6b 47 5a
39 98 d2 c7 72 a0 b6 2d
89 4c 69 f6 39 50 5b 83
fb 16 50 5f 35 4d 7b f2
e8 0b 28 ba 8f b3 a8 79
74 90 14 5d d2 cc 54 a9
3a 48 0a bb 69 66 2a c1
1d 24 05 c8 a1 33 15 f5
9b 12 97 64 c5 8c 9f ef
d8 09 de 32 f7 46 da e2
6c 91 6f 19 ee 23 6d 71
b6 35 f4 95 5d c8 d9 9e
5b 8f 7a df bb 64 f9 4f
b8 d2 3d fa c8 32 e9 b2
5c 69 8b 7d 64 19 e1 59
2e a1 d0 ab 32 99 e5 b9
17 c5 68 c0 19 d9 e7 c9
9e f7 34 60 99 f9 e6 f1
4f ee 1a 30 d9 e9 73 ed
fd 8c 47 95 8b fc 02 ce
eb 46 b6 df d0 7e 01 67
e0 23 5b fa 68 3f 95 a6
70 84 b8 7d 34 8a df 53
38 42 5c ab 1a 45 fa bc
1c 21 2e c0 0d b7 7d 5e
0e 85 17 60 93 ce ab 2f
07 d7 9e 30 dc 67 c0 82
84 79 6c 83 46 1c 60 c3
42 a9 36 d4 23 0e 30 f4
21 c1 1b 6a 84 07 18 7a
85 f5 98 35 42 96 0c 3d
d7 ef 4c 8f 21 4b 06 8b
fe e2 26 d2 85 b0 03 d0
7f 71 13 69 d7 58 94 68
aa ad 9c a1 fe 2c 4a 34
f7 4a 40 c8 ee 28 41 a2
ee 25 20 64 77 14 b5 51
77 87 10 32 ae 0a cf bd
ae d6 08 19 57 05 f2 cb
57 6b 04 99 be 97 79 f0
be a0 02 d9 5f de a9 78
5f 50 01 f9 ba 6f c1 3c
ba 28 95 e9 5d a2 f5 1e
91 93 c7 35 47 83 a8 24
dd dc f6 8f b6 d4 54 12
fb 6e 7b d2 5b 6a 2a 09
e8 37 a8 69 b8 35 15 91
74 8e 54 a1 5c 8f 9f dd
3a 47 2a c5 2e d2 da fb
1d b6 15 f7 17 69 6d e8
9b 5b 9f ee 9e a1 a3 74
45 78 cf 91 c3 32 88 2e

```

Base 6.4 vs Base 6.3

```

b7 3c f2 dd f4 19 44 17
ce 1e 79 fb 7a 99 22 9e
67 0f a9 e8 3d d9 11 4f
a6 92 c1 74 8b f9 9d b2
53 49 f5 3a d0 e9 db 59
bc b1 ef 1d 68 e1 f8 b9
5e cd e2 9b 34 e5 7c c9
42 ef c9 f9 d2 40 a7 65
21 e2 f1 e9 69 20 c6 a7
85 71 ed e1 a1 10 63 c6
d7 ad e3 e5 c5 08 a4 63
fe c3 e4 e7 f7 04 52 a4
7f f4 72 e6 ee 02 29 52
aa 7a 39 73 77 01 81 29
55 3d 89 ac ae 95 d5 81
a8 da 0d 95 39 83 24 51
54 6d 93 df 89 d4 12 bd
2a a3 dc fa d1 6a 09 cb
15 c4 6e 7d fd 35 91 f0
9f 62 37 ab eb 8f dd 78
da 31 8e c0 e0 d2 fb 3c
6d 8d 47 60 70 69 e8 1e
a3 d3 b6 30 38 a1 74 0f
4c 69 76 af 94 4e 0a cc
26 a1 3b c2 4a 27 05 66
13 c5 88 61 25 86 97 33
9c f7 44 a5 87 43 de 8c
4e ee 22 c7 d6 b4 6f 46
27 77 11 f6 6b 5a a2 23
86 ae 9d 7b a0 2d 51 84
43 57 db a8 50 83 bd 42
d4 03 28 29 2e dd 36 39
6a 94 14 81 17 fb 1b 89
35 4a 0a d5 9e e8 98 d1
8f 25 05 ff 4f 74 4c fd
d2 87 97 ea b2 3a 26 eb
69 d6 de 75 59 1d 13 e0
a1 6b 6f af b9 9b 9c 70
c5 a0 a2 c2 c9 d8 4e 38
ef b7 8b c2 7d 64 1c 4b
e2 ce d0 61 ab 32 0e b0
71 67 68 a5 c0 19 07 58
ad a6 34 c7 60 99 96 2c
c3 53 1a f6 30 d9 4b 16
f4 bc 0d 7b 18 f9 b0 0b
7a 5e 93 a8 0c e9 58 90
3d 2f dc 54 06 e1 2c 48
9a 1f ae 50 f5 3c 03 7b
4d 9a 57 28 ef 1e 94 a8
b3 4d be 14 e2 0f 4a 54
cc b3 5f 0a 71 92 25 2a
66 cc ba 05 ad 49 87 15
33 66 5d 97 c3 b1 d6 9f
8c 33 bb de f4 cd 6b da
46 8c c8 6f 7a f3 a0 6d
cf 94 a2 25 9a 04 3e a0
f2 4a 51 87 4d 02 1f 50
79 25 bd d6 b3 01 9a 28
a9 87 cb 6b cc 95 4d 14
c1 d6 f0 a0 66 df b3 0a
f5 6b 78 50 33 fa cc 05
ef a0 3c 28 8c 7d 66 97
e2 50 1e 14 46 ab 33 de
da c2 cd d4 0f 4d 10 af
6d 61 f3 6a 92 b3 08 c2

```

Base 6.4 vs Base 6.3

```

a3 a5 ec 35 49 cc 04 61
c4 c7 76 8f b1 66 02 a5
62 f6 3b d2 cd 33 01 c7
31 7b 88 69 f3 8c 95 f6
8d a8 44 a1 ec 46 df 7b
d3 54 22 c5 76 23 fa a8
d4 34 05 19 f4 b1 31 23
6a 1a 97 99 7a cd 8d 84
35 0d de d9 3d f3 d3 42
8f 93 6f f9 8b ec fc 21
d2 dc a2 e9 d0 76 7e 85
69 6e 51 e1 68 3b 3f d7
a1 37 bd e5 34 88 8a fe
c5 8e cb e7 1a 44 45 7f
d8 9a bb 18 11 63 06 4b
6c 4d c8 0c 9d a4 03 b0
36 b3 64 06 db 52 94 58
1b cc 32 03 f8 29 4a 2c
98 66 19 94 7c 81 25 16
4c 33 99 4a 3e d5 87 0b
26 8c d9 25 1f ff d6 90
13 46 f9 87 9a ea 6b 48
0d 92 1e 86 bc 25 4f f1
93 49 0f 43 5e 87 b2 ed
dc b1 92 b4 2f d6 59 e3
6e cd 49 5a 82 6b b9 e4
37 f3 b1 2d 41 a0 c9 72
8e ec cd 83 b5 50 f1 39
47 76 f3 d4 cf 28 ed 89
b6 3b ec 6a f2 14 e3 d1
3c c0 dc 12 69 7f 9d d3
1e 60 6e 09 a1 aa db fc
0f 30 37 91 c5 55 f8 7e
92 18 8e dd f7 bf 7c 3f
49 0c 47 fb ee ca 3e 8a
b1 06 b6 e8 77 65 1f 45
cd 03 5b 74 ae a7 9a b7
f3 94 b8 3a 57 c6 4d ce
7c e4 70 d5 d7 03 76 04
3e 72 38 ff fe 94 3b 02
1f 39 1c ea 7f 4a 88 01
9a 89 0e 75 aa 25 44 95
4d d1 07 af 55 87 22 df
b3 fd 96 c2 bf d6 11 fa
cc eb 4b 61 ca 6b 9d 7d
66 e0 b0 a5 65 a0 db ab
9e 8b 94 ad b4 4e 7a 87
4f d0 4a c3 5a 27 3d d6
b2 68 25 f4 2d 86 8b 6b
59 34 87 7a 83 43 d0 a0
b9 1a d6 3d d4 b4 68 50
c9 0d 6b 8b 6a 5a 34 28
f1 93 a0 d0 35 2d 1a 14
ed dc 50 68 8f 83 0d 0a
72 25 da 4d 24 52 dd 2f
39 87 6d b3 12 29 fb 82
89 d6 a3 cc 09 81 e8 41
d1 6b c4 66 91 d5 74 b5
fd a0 62 33 dd ff 3a cf
eb 50 31 8c fb ea 1d f2
e0 28 8d 46 e8 75 9b 79
70 14 d3 23 74 af d8 a9
a5 47 7f a4 fc 16 ea ef
c7 b6 aa 52 7e 0b 75 e2
f6 5b 55 29 3f 90 af 71

```

Base 6.4 vs Base 6.3

```

7b b8 bf 81 8a 48 c2 ad
a8 5c ca d5 45 24 61 c3
54 2e 65 ff b7 12 a5 f4
2a 17 a7 ea ce 09 c7 7a
15 9e c6 75 67 91 f6 3d
7f c5 1d c4 4d b0 d8 1f
aa f7 9b 62 b3 58 6c 9a
55 ee d8 31 cc 2c 36 4d
bf 77 6c 8d 66 16 1b b3
ca ae 36 d3 33 0b 98 cc
65 57 1b fc 8c 90 4c 66
a7 be 98 7e 46 48 26 33
c6 5f 4c 3f 23 24 13 8c
eb 5e ac 63 52 23 b6 3c
e0 2f 56 a4 29 84 5b 1e
70 82 2b 52 81 42 b8 0f
38 41 80 29 d5 21 5c 92
1c b5 40 81 ff 85 2e 49
0e cf 20 d5 ea d7 17 b1
07 f2 10 ff 75 fe 9e cd
96 79 08 ea af 7f 4f f3
5a b3 8a 56 7e b9 6b f4
2d cc 45 2b 3f c9 a0 7a
83 66 b7 80 8a f1 50 3d
d4 33 ce 40 45 ed 28 8b
6a 8c 67 20 b7 e3 14 d0
35 46 a6 10 ce e4 0a 68
8f 23 53 08 67 72 05 34
d2 84 bc 04 a6 39 97 1a
02 de c1 cf 3e ad f7 0e
01 6f f5 f2 1f c3 ee 07
95 a2 ef 79 9a f4 77 96
df 51 e2 a9 4d 7a ae 4b
fa bd 71 c1 b3 3d 57 b0
7d cb ad f5 cc 8b be 58
ab f0 c3 ef 66 d0 5f 2c
c0 78 f4 e2 33 68 ba 16
5f 11 18 bf cb 75 94 d2
ba 9d 0c ca f0 af 4a 69
5d db 06 65 78 c2 25 a1
bb f8 03 a7 3c 61 87 c5
c8 7c 94 c6 1e a5 d6 f7
64 3e 4a 63 0f c7 6b ee
32 1f 25 a4 92 f6 a0 77
19 9a 87 52 49 7b 50 ae
5c af 53 86 0d 3c 83 e8
2e c2 bc 43 93 1e d4 74
17 61 5e b4 dc 0f 6a 3a
9e a5 2f 5a 6e 92 35 1d
4f c7 82 2d 37 49 8f 9b
b2 f6 41 83 8e b1 d2 d8
59 7b b5 d4 47 cd 69 6c
b9 a8 cf 6a b6 f3 a1 36
b6 5c 43 14 11 e8 6e 53
5b 2e b4 0a 9d 74 37 bc
b8 17 5a 05 db 3a 8e 5e
5c 9e 2d 97 f8 1d 47 2f
2e 4f 83 de 7c 9b b6 82
17 b2 d4 6f 3e d8 5b 41
9e 59 6a a2 1f 6c b8 b5
4f b9 35 51 9a 36 5c cf
94 3b c6 d9 ab 4b cf ce
4a 88 63 f9 c0 b0 f2 67
25 44 a4 e9 60 58 79 a6
87 22 52 e1 30 2c a9 53

```

Base 6.4 vs Base 6.3

```

d6 11 29 e5 18 16 c1 bc
6b 9d 81 e7 0c 0b f5 5e
a0 db d5 e6 06 90 ef 2f
50 f8 ff 73 03 48 e2 82
11 9a 58 81 f4 3b 30 c8
9d 4d 2c d5 7a 88 18 64
db b3 16 ff 3d 44 0c 32
f8 cc 0b ea 8b 22 06 19
7c 66 90 75 d0 11 03 99
3e 33 48 af 68 9d 94 d9
1f 8c 24 c2 34 db 4a f9
9a 46 12 61 1a f8 25 e9
eb 4a 3d 85 45 cd f9 03
e0 25 8b d7 b7 f3 e9 94
70 87 d0 fe ce ec e1 4a
38 d6 68 7f 67 76 e5 25
1c 6b 34 aa a6 3b e7 87
0e a0 1a 55 53 88 e6 d6
07 50 0d bf bc 44 73 6b
96 28 93 ca 5e 22 ac a0
4e 22 6c 41 90 f6 54 f4
27 11 36 b5 48 7b 2a 7a
86 9d 1b cf 24 a8 15 3d
43 db 98 f2 12 54 9f 8b
b4 f8 4c 79 09 2a da d0
5a 7c 26 a9 91 15 6d 68
2d 3e 13 c1 dd 9f a3 34
83 1f 9c f5 fb da c4 1a
1e c9 11 ac f0 01 94 eb
0f f1 9d 56 78 95 4a e0
92 ed db 2b 3c df 25 70
49 e3 f8 80 1e fa 87 38
b1 e4 7c 40 0f 7d d6 1c
cd 72 3e 20 92 ab 6b 0e
f3 39 1f 10 49 c0 a0 07
ec 89 9a 08 b1 60 50 96
97 0d 9d ae 55 db 2c 02
de 93 db 57 bf f8 16 01
6f dc f8 be ca 7c 0b 95
a2 6e 7c 5f 65 3e 90 df
51 37 3e ba a7 1f 48 fa
bd 8e 1f 5d c6 9a 24 7d
cb 47 9a bb 63 4d 12 ab
f0 b6 4d c8 a4 b3 09 c0
73 79 06 2b 31 1b 2b 73
ac a9 03 80 8d 98 80 ac
56 c1 94 40 d3 4c 40 56
2b f5 4a 20 fc 26 20 2b
80 ef 25 10 7e 13 10 80
40 e2 87 08 3f 9c 08 40
20 71 d6 04 8a 4e 04 20
10 ad 6b 02 45 27 02 10
2c f3 e7 64 86 a1 fb 86
16 ec e6 32 43 c5 e8 43
0b 76 73 19 b4 f7 74 b4
90 3b ac 99 5a ee 3a 5a
48 88 56 d9 2d 77 1d 2d
24 44 2b f9 83 ae 9b 83
12 22 80 e9 d4 57 d8 d4
09 11 40 e1 6a be 6c 6a
eb a5 44 9e 44 d9 5f 6e
e0 c7 22 4f 22 f9 ba 37
70 f6 11 b2 11 e9 5d 8e
38 7b 9d 59 9d e1 bb 47
1c a8 db b9 db e5 c8 b6

```

Base 6.4 vs Base 6.3

```

0e 54 f8 c9 f8 e7 64 5b
07 2a 7c f1 7c e6 32 b8
96 15 3e ed 3e 73 19 5c
a1 5b 77 40 84 50 39 f4
c5 b8 ae 20 42 28 89 7a
f7 5c 57 10 21 14 d1 3d
ee 2e be 08 85 0a fd 8b
77 17 5f 04 d7 05 eb d0
ae 9e ba 02 fe 97 e0 68
57 4f 5d 01 7f de 70 34
be b2 bb 95 aa 6f 38 1a
4a 46 7c 95 c7 4c dc 90
25 23 3e df f6 26 6e 48
87 84 1f fa 7b 13 37 24
d6 42 9a 7d a8 9c 8e 12
6b 21 4d ab 54 4e 47 09
a0 85 b3 c0 2a 27 b6 91
50 d7 cc 60 15 86 5b dd
28 fe 66 30 9f 43 b8 fb
53 52 40 d2 86 a6 ef 64
bc 29 20 69 43 53 e2 32
5e 81 10 a1 b4 bc 71 19
2f d5 08 c5 5a 5e ad 99
82 ff 04 f7 2d 2f c3 d9
41 ea 02 ee 83 82 f4 f9
b5 75 01 77 d4 41 7a e9
cf af 95 ae 6a b5 3d e1
64 41 9a b0 a1 36 14 52
32 b5 4d 58 c5 1b 0a 29
19 cf b3 2c f7 98 05 81
99 f2 cc 16 ee 4c 97 d5
d9 79 66 0b 77 26 de ff
f9 a9 33 90 ae 13 6f ea
e9 c1 8c 48 57 9c a2 75
e1 f5 46 24 be 4e 51 af
cd 26 92 2d 7f ce 6e d8
f3 13 49 83 aa 67 37 6c
ec 9c b1 d4 55 a6 8e 36
76 4e cd 6a bf 53 47 1b
3b 27 f3 35 ca bc b6 98
88 86 ec 8f 65 5e 5b 4c
44 43 76 d2 a7 2f b8 26
22 b4 3b 69 c6 82 5c 13
e9 22 12 b0 a3 9f f2 7d
e1 11 09 58 c4 da 79 ab
e5 9d 91 2c 62 6d a9 c0
e7 db dd 16 31 a3 c1 60
e6 f8 fb 0b 8d c4 f5 30
73 7c e8 90 d3 62 ef 18
ac 3e 74 48 fc 31 e2 0c
56 1f 3a 24 7e 8d 71 06
a7 dd 8e 26 a4 7f b0 26
c6 fb 47 13 52 aa 58 13
63 e8 b6 9c 29 55 2c 9c
a4 74 5b 4e 81 bf 16 4e
52 3a b8 27 d5 ca 0b 27
29 1d 5c 86 ff 65 90 86
81 9b 2e 43 ea a7 48 43
d5 d8 17 b4 75 c6 24 b4
64 e4 48 1d 42 68 8b cd
32 72 24 9b 21 34 d0 f3
19 39 12 d8 85 1a 68 ec
99 89 09 6c d7 0d 34 76
d9 d1 91 36 fe 93 1a 3b
f9 fd dd 1b 7f dc 0d 88

```

Base 6.4 vs Base 6.3

```

e9 eb fb 98 aa 6e 93 44
e1 e0 e8 4c 55 37 dc 22
68 f4 3f ce 21 51 f1 d8
34 7a 8a 67 85 bd ed 6c
1a 3d 45 a6 d7 cb e3 36
0d 8b b7 53 fe f0 e4 1b
93 d0 ce bc 7f 78 72 98
dc 68 67 5e aa 3c 39 4c
6e 34 a6 2f 55 1e 89 26
37 1a 53 82 bf 0f d1 13
03 35 48 d6 67 5a c5 62
94 8f 24 6b a6 2d f7 31
4a d2 12 a0 53 83 ee 8d
25 69 09 50 bc d4 77 d3
87 a1 91 28 5e 6a ae fc
d6 c5 dd 14 2f 35 57 7e
6b f7 fb 0a 82 8f be 3f
a0 ee e8 05 41 d2 5f 8a
61 f0 ff 33 0f 06 b5 bb
a5 78 ea 8c 92 03 cf c8
c7 3c 75 46 49 94 f2 64
f6 1e af 23 b1 4a 79 32
7b 0f c2 84 cd 25 a9 19
a8 92 61 42 f3 87 c1 99
54 49 a5 21 ec d6 f5 d9
2a b1 c7 85 76 6b ef f9
80 a0 6a 7f b0 01 d5 3e
40 50 35 aa 58 95 ff 1f
20 28 8f 55 2c df ea 9a
10 14 d2 bf 16 fa 75 4d
08 0a 69 ca 0b 7d af b3
04 05 a1 65 90 ab c2 cc
02 97 c5 a7 48 c0 61 66
01 de f7 c6 24 60 a5 33
07 c7 39 17 3f b2 a3 2d
96 f6 89 9e 8a 59 c4 83
4b 7b d1 4f 45 b9 62 d4
b0 a8 fd b2 b7 c9 31 6a
58 54 eb 59 ce f1 8d 35
2c 2a e0 b9 67 ed d3 8f
16 15 70 c9 a6 e3 fc d2
0b 9f 38 f1 53 e4 7e 69
ba 0a bb 42 2b 4f e5 34
5d 05 c8 21 80 b2 e7 1a
bb 97 64 85 40 59 e6 0d
c8 de 32 d7 20 b9 73 93
64 6f 19 fe 10 c9 ac dc
32 a2 99 7f 08 f1 56 6e
19 51 d9 aa 04 ed 2b 37
99 bd f9 55 02 e3 80 8e
b9 75 34 98 73 b1 89 74
c9 af 1a 4c ac cd d1 3a
f1 c2 0d 26 56 f3 fd 1d
ed 61 93 13 2b ec eb 9b
e3 a5 dc 9c 80 76 e0 d8
e4 c7 6e 4e 40 3b 70 6c
72 f6 37 27 20 88 38 36
39 7b 8e 86 10 44 1c 1b
92 42 c7 94 d8 1e 1e cf
49 21 f6 4a 6c 0f 0f f2
b1 85 7b 25 36 92 92 79
cd d7 a8 87 1b 49 49 a9
f3 fe 54 d6 98 b1 b1 c1
ec 7f 2a 6b 4c cd cd f5
76 aa 15 a0 26 f3 f3 ef

```

Base 6.4 vs Base 6.3

```

3b 55 9f 50 13 ec ec e2
c0 c0 47 5a 0b 47 b4 95
60 60 b6 2d 90 b6 5a df
30 30 5b 83 48 5b 2d fa
18 18 b8 d4 24 b8 83 7d
0c 0c 5c 6a 12 5c d4 ab
06 06 2e 35 09 2e 6a c0
03 03 17 8f 91 17 35 60
94 94 9e d2 dd 9e 8f 30
a1 29 3f 6a 66 75 d3 ae
c5 81 8a 35 33 af fc 57
f7 d5 45 8f 8c c2 7e be
ee ff b7 d2 46 61 3f 5f
77 ea ce 69 23 a5 8a ba
ae 75 67 a1 84 c7 45 5d
57 af a6 c5 42 f6 b7 bb
be c2 53 f7 21 7b ce c8
38 0e 56 77 e2 a6 86 90
1c 07 2b ae 71 53 43 48
0e 96 80 57 ad bc b4 24
07 4b 40 be c3 5e 5a 12
96 b0 20 5f f4 2f 2d 09
4b 58 10 ba 7a 82 83 91
b0 2c 08 5d 3d 41 d4 dd
58 16 04 bb 8b b5 6a fb
9b e5 90 77 c2 37 2a 8b
d8 e7 48 ae 61 8e 15 d0
6c e6 24 57 a5 47 9f 68
36 73 12 be c7 b6 da 34
1b ac 09 5f f6 5b 6d 1a
98 56 91 ba 7b b8 a3 0d
4c 2b dd 5d a8 5c c4 93
26 80 fb bb 54 2e 62 dc
e0 c2 7b c7 a2 6c 83 c9
70 61 a8 f6 51 36 d4 f1
38 a5 54 7b bd 1b 6a ed
1c c7 2a a8 cb 98 35 e3
0e f6 15 54 f0 4c 8f e4
07 7b 9f 2a 78 26 d2 72
96 a8 da 15 3c 13 69 39
4b 54 6d 9f 1e 9c a1 89
c0 e1 26 a0 d7 11 77 7a
60 e5 13 50 fe 9d ae 3d
30 e7 9c 28 7f db 57 8b
18 e6 4e 14 aa f8 be d0
0c 73 27 0a 55 7c 5f 68
06 ac 86 05 bf 3e ba 34
03 56 43 97 ca 1f 5d 1a
94 2b b4 de 65 9a bb 0d
80 48 c5 b6 30 b6 3c ae
40 24 f7 5b 18 5b 1e 57
20 12 ee b8 0c b8 0f be
10 09 77 5c 06 5c 92 5f
08 91 ae 2e 03 2e 49 ba
04 dd 57 17 94 17 b1 5d
02 fb be 9e 4a 9e cd bb
01 e8 5f 4f 25 4f f3 c8
ef 68 f0 97 88 5b 33 2d
e2 34 78 de 44 b8 8c 83
71 1a 3c 6f 22 5c 46 d4
ad 0d 1e a2 11 2e 23 6a
c3 93 0f 51 9d 17 84 35
f4 dc 92 bd db 9e 42 8f
7a 6e 49 cb f8 4f 21 d2
3d 37 b1 f0 7c b2 85 69

```

Base 6.4 vs Base 6.3

```

45 64 fb a5 ca 13 65 7b
b7 32 e8 c7 65 9c a7 a8
ce 19 74 f6 a7 4e c6 54
67 99 3a 7b c6 27 63 2a
a6 d9 1d a8 63 86 a4 15
53 f9 9b 54 a4 43 52 9f
bc e9 d8 2a 52 b4 29 da
5e e1 6c 15 29 5a 81 6d
5e db fd f0 f3 ad f2 65
2f f8 eb 78 ec c3 79 a7
82 7c e0 3c 76 f4 a9 c6
41 3e 70 1e 3b 7a c1 63
b5 1f 38 0f 88 3d f5 a4
cf 9a 1c 92 44 8b ef 52
f2 4d 0e 49 22 d0 e2 29
79 b3 07 b1 11 68 71 81
43 22 e2 6b e6 1b 79 81
b4 11 71 a0 73 98 a9 d5
5a 9d ad 50 ac 4c c1 ff
2d db c3 28 56 26 f5 ea
83 f8 f4 14 2b 13 ef 75
d4 7c 7a 0a 80 9c e2 af
6a 3e 3d 05 40 4e 71 c2
35 1f 8b 97 20 27 ad 61
b0 99 61 74 51 1c 8d 38
58 d9 a5 3a bd 0e d3 1c
2c f9 c7 1d cb 07 fc 0e
16 e9 f6 9b f0 96 7e 07
0b e1 7b d8 78 4b 3f 96
90 e5 a8 6c 3c b0 8a 4b
48 e7 54 36 1e 58 45 b0
24 e6 2a 1b 0f 2c b7 58
f9 42 f4 31 f5 05 21 93
e9 21 7a 8d ef 97 85 dc
e1 85 3d d3 e2 de d7 6e
e5 d7 8b fc 71 6f fe 37
e7 fe d0 7e ad a2 7f 8e
e6 7f 68 3f c3 51 aa 47
73 aa 34 8a f4 bd 55 b6
ac 55 1a 45 7a cb bf 5b
63 41 4d d4 73 10 22 4c
a4 b5 b3 6a ac 08 11 26
52 cf cc 35 56 04 9d 13
29 f2 66 8f 2b 02 db 9c
81 79 33 d2 80 01 f8 4e
d5 a9 8c 69 40 95 7c 27
ff c1 46 a1 20 df 3e 86
ea f5 23 c5 10 fa 1f 43
b0 c2 5a 82 c0 14 b8 ec
58 61 2d 41 60 0a 5c 76
2c a5 83 b5 30 05 2e 3b
16 c7 d4 cf 18 97 17 88
0b f6 6a f2 0c de 9e 44
90 7b 35 79 06 6f 4f 22
48 a8 8f a9 03 a2 b2 11
24 54 d2 c1 94 51 59 9d
a2 79 02 a0 fd 30 f5 93
51 a9 01 50 eb 18 ef dc
bd c1 95 28 e0 0c e2 6e
cb f5 df 14 70 06 71 37
f0 ef fa 0a 38 03 ad 8e
78 e2 7d 05 1c 94 c3 47
3c 71 ab 97 0e 4a f4 b6
1e ad c0 de 07 25 7a 5b
3c 8b b5 b8 f2 43 6c 2b

```

Base 6.4 vs Base 6.3

```

1e d0 cf 5c 79 b4 36 80
0f 68 f2 2e a9 5a 1b 40
92 34 79 17 c1 2d 98 20
49 1a a9 9e f5 83 4c 10
b1 0d c1 4f ef d4 26 08
cd 93 f5 b2 e2 6a 13 04
f3 dc ef 59 71 35 9c 02
37 8d da 4e eb f2 8e 04
8e d3 6d 27 e0 79 47 02
47 fc a3 86 70 a9 b6 01
b6 7e c4 43 38 c1 5b 95
5b 3f 62 b4 1c f5 b8 df
b8 8a 31 5a 0e ef 5c fa
5c 45 8d 2d 07 e2 2e 7d
2e b7 d3 83 96 71 17 ab
37 67 24 51 bc 8d 48 8a
8e a6 12 bd 5e d3 24 45
47 53 09 cb 2f fc 12 b7
b6 bc 91 f0 82 7e 09 ce
5b 5e dd 78 41 3f 91 67
b8 2f fb 3c b5 8a dd a6
5c 82 e8 1e cf 45 fb 53
2e 41 74 0f f2 b7 e8 bc
dd 99 3b 06 c3 4b c6 8a
fb d9 88 03 f4 b0 63 45
e8 f9 44 94 7a 58 a4 b7
74 e9 22 4a 3d 2c 52 ce
3a e1 11 25 8b 16 29 67
1d e5 9d 87 d0 0b 81 a6
9b e7 db d6 68 90 d5 53
d8 e6 f8 6b 34 48 ff bc
f2 f1 7b a8 6b 8b dc 17
79 ed a8 54 a0 d0 6e 9e
a9 e3 54 2a 50 68 37 4f
c1 e4 2a 15 28 34 8e b2
f5 72 15 9f 14 1a 47 59
ef 39 9f da 0a 0d b6 b9
e2 89 da 6d 05 93 5b c9
71 d1 6d a3 97 dc b8 f1
d6 4b dc 4c 1b 70 4f c2
6b b0 6e 26 98 38 b2 61
a0 58 37 13 4c 1c 59 a5
50 2c 8e 9c 26 0e b9 c7
28 16 47 4e 13 07 c9 f6
14 0b b6 27 9c 96 f1 7b
0a 90 5b 86 4e 4b ed a8
05 48 b8 43 27 b0 e3 54
26 93 39 76 b6 9f 7d 99
13 dc 89 3b 5b da ab d9
9c 6e d1 88 b8 6d c0 f9
4e 37 fd 44 5c a3 60 e9
27 8e eb 22 2e c4 30 e1
86 47 e0 11 17 62 18 e5
43 b6 70 9d 9e 31 0c e7
b4 5b 38 db 4f 8d 06 e6
58 96 a0 7d af 83 e4 89
2c 4b 50 ab c2 d4 72 d1
16 b0 28 c0 61 6a 39 fd
0b 58 14 60 a5 35 89 eb
90 2c 0a 30 c7 8f d1 e0
48 16 05 18 f6 d2 fd 70
24 0b 97 0c 7b 69 eb 38
12 90 de 06 a8 a1 e0 1c
a6 24 3d 9f 62 a0 10 dc
53 12 8b da 31 50 08 6e

```

Base 6.4 vs Base 6.3

```

bc 09 d0 6d 8d 28 04 37
5e 91 68 a3 d3 14 02 8e
2f dd 34 c4 fc 0a 01 47
82 fb 1a 62 7e 05 95 b6
41 e8 0d 31 3f 97 df 5b
b5 74 93 8d 8a de fa b8
48 3f 0f 0e ae 89 3c a4
24 8a 92 07 57 d1 1e 52
12 45 49 96 be fd 0f 29
09 b7 b1 4b 5f eb 92 81
91 ce cd b0 ba e0 49 d5
dd 67 f3 58 5d 70 b1 ff
fb a6 ec 2c bb 38 cd ea
e8 53 76 16 c8 1c f3 75
ab f1 0c 3a 25 c4 41 b6
c0 ed 06 1d 87 62 b5 5b
60 e3 03 9b d6 31 cf b8
30 e4 94 d8 6b 8d f2 5c
18 72 4a 6c a0 d3 79 2e
0c 39 25 36 50 fc a9 17
06 89 87 1b 28 7e c1 9e
03 d1 d6 98 14 3f f5 4f
33 d0 f0 e7 86 e8 3a 77
8c 68 78 e6 43 74 1d ae
46 34 3c 73 b4 3a 9b 57
23 1a 1e ac 5a 1d d8 be
84 0d 0f 56 2d 9b 6c 5f
42 93 92 2b 83 d8 36 ba
21 dc 49 80 d4 6c 1b 5d
85 6e b1 40 6a 36 98 bb
43 c6 08 15 6a 16 24 05
b4 63 04 9f 35 0b 12 97
5a a4 02 da 8f 90 09 de
2d 52 01 6d d2 48 91 6f
83 29 95 a3 69 24 dd a2
d4 81 df c4 a1 12 fb 51
6a d5 fa 62 c5 09 e8 bd
35 ff 7d 31 f7 91 74 cb
54 73 1f f8 5c 41 09 38
2a ac 9a 7c 2e b5 91 1c
15 56 4d 3e 17 cf dd 0e
9f 2b b3 1f 9e f2 fb 07
da 80 cc 9a 4f 79 e8 96
6d 40 66 4d b2 a9 74 4b
a3 20 33 b3 59 c1 3a b0
c4 10 8c cc b9 f5 1d 58
38 2d 1c 17 df 3c 80 66
1c 83 0e 9e fa 1e 40 33
0e d4 07 4f 7d 0f 20 8c
07 6a 96 b2 ab 92 10 46
96 35 4b 59 c0 49 08 23
4b 8f b0 b9 60 b1 04 84
b0 d2 58 c9 30 cd 02 42
58 69 2c f1 18 f3 01 21
b8 af f0 4a 58 31 dc 8b
5c c2 78 25 2c 8d 6e d0
2e 61 3c 87 16 d3 37 68
17 a5 1e d6 0b fc 8e 34
9e c7 0f 6b 90 7e 47 1a
4f f6 92 a0 48 3f b6 0d
b2 7b 49 50 24 8a 5b 93
59 a8 b1 28 12 45 b8 dc
9d ee eb 6a 6b 0a ac a6
db 77 e0 35 a0 05 56 53
f8 ae 70 8f 50 97 2b bc

```

Base 6.4 vs Base 6.3

```

7c 57 38 d2 28 de 80 5e
3e be 1c 69 14 6f 40 2f
1f 5f 0e a1 0a a2 20 82
9a ba 07 c5 05 51 10 41
4d 5d 96 f7 97 bd 08 b5
43 e2 34 c6 35 47 2b 94
b4 71 1a 63 8f b6 80 4a
5a ad 0d a4 d2 5b 40 25
2d c3 93 52 69 b8 20 87
83 f4 dc 29 a1 5c 10 d6
d4 7a 6e 81 c5 2e 08 6b
6a 3d 37 d5 f7 17 04 a0
35 8b 8e ff ee 9e 02 50
70 4f cc a7 0d 4e 98 38
38 b2 66 c6 93 27 4c 1c
1c 59 33 63 dc 86 26 0e
0e b9 8c a4 6e 43 13 07
07 c9 46 52 37 b4 9c 96
96 f1 23 29 8e 5a 4e 4b
4b ed 84 81 47 2d 27 b0
b0 e3 42 d5 b6 83 86 58
4e 81 42 0c 86 d1 67 3d
27 d5 21 06 43 fd a6 8b
86 ff 85 03 b4 eb 53 d0
43 ea d7 94 5a e0 bc 68
b4 75 fe 4a 2d 70 5e 34
5a af 7f 25 83 38 2f 1a
2d c2 aa 87 d4 1c 82 0d
83 61 55 d6 6a 0e 41 93
bd e7 5c ba d7 32 5d eb
cb e6 2e 5d fe 19 bb e0
f0 73 17 bb 7f 99 c8 70
78 ac 9e c8 aa d9 64 38
3c 56 4f 64 55 f9 32 1c
1e 2b b2 32 bf e9 19 0e
0f 80 59 19 ca e1 99 07
92 40 b9 99 65 e5 d9 96
29 a1 60 19 97 e5 9e 40
81 c5 30 99 de e7 4f 20
d5 f7 18 d9 6f e6 b2 10
ff ee 0c f9 a2 73 59 08
ea 77 06 e9 51 ac b9 04
75 ae 03 e1 bd 56 c9 02
af 57 94 e5 cb 2b f1 01
c2 be 4a e7 f0 80 ed 95
45 c1 42 73 ff d2 cb 88
b7 f5 21 ac ea 69 f0 44
ce ef 85 56 75 a1 78 22
67 e2 d7 2b af c5 3c 11
a6 71 fe 80 c2 f7 1e 9d
53 ad 7f 40 61 ee 0f db
bc c3 aa 20 a5 77 92 f8
5e f4 55 10 c7 ae 49 7c
fb 62 2b c5 32 03 01 65
e8 31 80 f7 19 94 95 a7
74 8d 40 ee 99 4a df c6
3a d3 20 77 d9 25 fa 63
1d fc 10 ae f9 87 7d a4
9b 7e 08 57 e9 d6 ab 52
d8 3f 04 be e1 6b c0 29
6c 8a 02 5f e5 a0 60 81
c2 4e 6e 92 13 b2 4e 9e
61 27 37 49 9c 59 27 4f
a5 86 8e b1 4e b9 86 b2
c7 43 47 cd 27 c9 43 59

```

Base 6.4 vs Base 6.3

```

f6 b4 b6 f3 86 f1 b4 b9
7b 5a 5b ec 43 ed 5a c9
a8 2d b8 76 b4 e3 2d f1
54 83 5c 3b 5a e4 83 ed
ae d0 20 f3 f5 0d 42 7c
57 68 10 ec ef 93 21 3e
be 34 08 76 e2 dc 85 1f
5f 1a 04 3b 71 6e d7 9a
ba 0d 02 88 ad 37 fe 4d
5d 93 01 44 c3 8e 7f b3
bb dc 95 22 f4 47 aa cc
c8 6e df 11 7a b6 55 66
ee 1f 42 cb b0 85 a2 91
77 9a 21 f0 58 d7 51 dd
ae 4d 85 78 2c fe bd fb
57 b3 d7 3c 16 7f cb e8
be cc fe 1e 0b aa f0 74
5f 66 7f 0f 90 55 78 3a
ba 33 aa 92 48 bf 3c 1d
5d 8c 55 49 24 ca 1e 9b
e7 be 59 48 27 c3 94 12
e6 5f b9 24 86 f4 4a 09
73 ba c9 12 43 7a 25 91
ac 5d f1 09 b4 3d 87 dd
56 bb ed 91 5a 8b d6 fb
2b c8 e3 dd 2d d0 6b e8
80 64 e4 fb 83 68 a0 74
40 32 72 e8 d4 34 50 3a
c1 f0 05 58 e1 ea 64 4e
f5 78 97 2c e5 75 32 27
ef 3c de 16 e7 af 19 86
e2 1e 6f 0b e6 c2 99 43
71 0f a2 90 73 61 d9 b4
ad 92 51 48 ac a5 f9 5a
c3 49 bd 24 56 c7 e9 2d
f4 b1 cb 12 2b f6 e1 83
44 03 c3 55 f8 e4 45 c7
22 94 f4 bf 7c 72 b7 f6
11 4a 7a ca 3e 39 ce 7b
9d 25 3d 65 1f 89 67 a8
db 87 8b a7 9a d1 a6 54
f8 d6 d0 c6 4d fd 53 2a
7c 6b 68 63 b3 eb bc 15
3e a0 34 a4 cc e0 5e 9f
ec 8b f3 17 37 cc 03 0c
76 d0 ec 9e 8e 66 94 06
3b 68 76 4f 47 33 4a 03
88 34 3b b2 b6 8c 25 94
44 1a 88 59 5b 46 87 4a
22 0d 44 b9 b8 23 d6 25
11 93 22 c9 5c 84 6b 87
9d dc 11 f1 2e 42 a0 d6
f2 df 52 4e 08 e0 93 c0
79 fa 29 27 04 70 dc 60
a9 7d 81 86 02 38 6e 30
c1 ab d5 43 01 1c 37 18
f5 c0 ff b4 95 0e 8e 0c
ef 60 ea 5a df 07 47 06
e2 30 75 2d fa 96 b6 03
71 18 af 83 7d 4b 5b 94
f8 62 3a 2f 70 3f 98 c2
7c 31 1d 82 38 8a 4c 61
3e 8d 9b 41 1c 45 26 a5
1f d3 d8 b5 0e b7 13 c7
9a fc 6c cf 07 ce 9c f6

```

Base 6.4 vs Base 6.3

```

4d 7e 36 f2 96 67 4e 7b
b3 3f 1b 79 4b a6 27 a8
cc 8a 98 a9 b0 53 86 54
96 e7 ad 84 7a e8 3e 25
4b e6 c3 42 3d 74 1f 87
b0 73 f4 21 8b 3a 9a d6
58 ac 7a 85 d0 1d 4d 6b
2c 56 3d d7 68 9b b3 a0
16 2b 8b fe 34 d8 cc 50
0b 80 d0 7f 1a 6c 66 28
90 40 68 aa 0d 36 33 14
4c 21 d2 d1 31 48 41 1a
26 85 69 fd 8d 24 b5 0d
13 d7 a1 eb d3 12 cf 93
9c fe c5 e0 fc 09 f2 dc
4e 7f f7 70 7e 91 79 6e
27 aa ee 38 3f dd a9 37
86 55 77 1c 8a fb c1 8e
43 bf ae 0e 45 e8 f5 47
9c a7 56 8c 28 96 e0 39
4e c6 2b 46 14 4b 70 89
27 63 80 23 0a b0 38 d1
86 a4 40 84 05 58 1c fd
43 52 20 42 97 2c 0e eb
b4 29 10 21 de 16 07 e0
5a 81 08 85 6f 0b 96 70
2d d5 04 d7 a2 90 4b 38
df 37 2c 50 9a 4a f9 fd
fa 8e 16 28 4d 25 e9 eb
7d 47 0b 14 b3 87 e1 e0
ab b6 90 0a cc d6 e5 70
c0 5b 48 05 66 6b e7 38
60 b8 24 97 33 a0 e6 1c
30 5c 12 de 8c 50 73 0e
18 2e 09 6f 46 28 ac 07
dd 36 fa 70 0c 36 31 c5
fb 1b 7d 38 06 1b 8d f7
e8 98 ab 1c 03 98 d3 ee
74 4c c0 0e 94 4c fc 77
3a 26 60 07 4a 26 7e ae
1d 13 30 96 25 13 3f 57
9b 9c 18 4b 87 9c 8a be
d8 4e 0c b0 d6 4e 45 5f
5d 30 0d 67 16 7c 93 17
bb 18 93 a6 0b 3e dc 9e
c8 0c dc 53 90 1f 6e 4f
64 06 6e bc 48 9a 37 b2
32 03 37 5e 24 4d 8e 59
19 94 8e 2f 12 b3 47 b9
99 4a 47 82 09 cc b6 c9
d9 25 b6 41 91 66 5b f1
fc 6a 5c da 62 b9 dc 3a
7e 35 2e 6d 31 c9 6e 1d
3f 8f 17 a3 8d f1 37 9b
8a d2 9e c4 d3 ed 8e d8
45 69 4f 62 fc e3 47 6c
b7 a1 b2 31 7e e4 b6 36
ce c5 59 8d 3f 72 5b 1b
67 f7 b9 d3 8a 39 b8 98
b7 0a dd bf 30 83 d9 aa
ce 05 fb ca 18 d4 f9 55
67 97 e8 65 0c 6a e9 bf
a6 de 74 a7 06 35 e1 ca
53 6f 3a c6 03 8f e5 65
bc a2 1d 63 94 d2 e7 a7

```

Base 6.4 vs Base 6.3

```

5e 51 9b a4 4a 69 e6 c6
2f bd d8 52 25 a1 73 63
17 cd 93 2d f3 bd 7b a7
9e f3 dc 83 ec cb a8 c6
4f ec 6e d4 76 f0 54 63
b2 76 37 6a 3b 78 2a a4
59 3b 8e 35 88 3c 15 52
b9 88 47 8f 44 1e 9f 29
c9 44 b6 d2 22 0f da 81
f1 22 5b 69 11 92 6d d5
14 da c3 2a 69 2b 13 5e
0a 6d f4 15 a1 80 9c 2f
05 a3 7a 9f c5 40 4e 82
97 c4 3d da f7 20 27 41
de 62 8b 6d ee 10 86 b5
6f 31 d0 a3 77 08 43 cf
a2 8d 68 c4 ae 04 b4 f2
51 d3 34 62 57 02 5a 79
57 32 ed e4 aa 80 3d e5
be 19 e3 72 55 40 8b e7
5f 99 e4 39 bf 20 d0 e6
ba d9 72 89 ca 10 68 73
5d f9 39 d1 65 08 34 ac
bb e9 89 fd a7 04 1a 56
c8 e1 d1 eb c6 02 0d 2b
64 e5 fd e0 63 01 93 80
2d 67 29 b5 4b cb 8a dd
83 a6 81 cf b0 f0 45 fb
d4 53 d5 f2 58 78 b7 e8
6a bc ff 79 2c 3c ce 74
35 5e ea a9 16 1e 67 3a
8f 2f 75 c1 0b 0f a6 1d
d2 82 af f5 90 92 53 9b
69 41 c2 ef 48 49 bc d8
aa 03 6b 86 1d e9 49 07
55 94 a0 43 9b e1 b1 96
bf 4a 50 b4 d8 e5 cd 4b
ca 25 28 5a 6c e7 f3 b0
65 87 14 2d 36 e6 ec 58
a7 d6 0a 83 1b 73 76 2c
c6 6b 05 d4 98 ac 3b 16
63 a0 97 6a 4c 56 88 0b
14 df b2 0a 98 a1 c6 1e
0a fa 59 05 4c c5 63 0f
05 7d b9 97 26 f7 a4 92
97 ab c9 de 13 ee 52 49
de c0 f1 6f 9c 77 29 b1
6f 60 ed a2 4e ae 81 cd
a2 30 e3 51 27 57 d5 f3
51 18 e4 bd 86 be ff ec
52 43 cd 15 20 55 7d e5
29 b4 f3 9f 10 bf ab e7
81 5a ec da 08 ca c0 e6
d5 2d 76 6d 04 65 60 73
ff 83 3b a3 02 a7 30 ac
ea d4 88 c4 01 c6 18 56
75 6a 44 62 95 63 0c 2b
af 35 22 31 df a4 06 80
a0 a4 a3 c3 61 e5 d8 3b
50 52 c4 f4 a5 e7 6c 88
28 29 62 7a c7 e6 36 44
14 81 31 3d f6 73 1b 22
0a d5 8d 8b 7b ac 98 11
05 ff d3 d0 a8 56 4c 9d
97 ea fc 68 54 2b 26 db

```

Base 6.4 vs Base 6.3

```

de 75 7e 34 2a 80 13 f8
60 fa 01 a5 41 ec 5e f9
30 7d 95 c7 b5 76 2f e9
18 ab df f6 cf 3b 82 e1
0c c0 fa 7b f2 88 41 e5
06 60 7d a8 79 44 b5 e7
03 30 ab 54 a9 22 cf e6
94 18 c0 2a c1 11 f2 73
4a 0c 60 15 f5 9d 79 ac
5f 36 02 e4 69 ab da 57
ba 1b 01 72 a1 c0 6d be
5d 98 95 39 c5 60 a3 5f
bb 4c df 89 f7 30 c4 ba
c8 26 fa d1 ee 18 62 5d
64 13 7d fd 77 0c 31 bb
32 9c ab eb ae 06 8d c8
19 4e c0 e0 57 03 d3 64
7b b5 08 24 d3 72 06 e8
a8 cf 04 12 fc 39 03 74
54 f2 02 09 7e 89 94 3a
2a 79 01 91 3f d1 4a 1d
15 a9 95 dd 8a fd 25 9b
9f c1 df fb 45 eb 87 d8
da f5 fa e8 b7 e0 d6 6c
6d ef 7d 74 ce 70 6b 36
b2 c0 c9 d4 5c d2 5e b3
59 60 f1 6a 2e 69 2f cc
b9 30 ed 35 17 a1 82 66
c9 18 e3 8f 9e c5 41 33
f1 0c e4 d2 4f f7 b5 8c
ed 06 72 69 b2 ee cf 46
e3 03 39 a1 59 77 f2 23
e4 94 89 c5 b9 ae 79 84
21 3a 83 bd 5d 54 30 41
85 1d d4 cb bb 2a 18 b5
d7 9b 6a f0 c8 15 0c cf
fe d8 35 78 64 9f 06 f2
7f 6c 8f 3c 32 da 03 79
aa 36 d2 1e 19 6d 94 a9
55 1b 69 0f 99 a3 4a c1
bf 98 a1 92 d9 c4 25 f5
8c 1f c7 eb fd 22 f4 bd
46 9a f6 e0 eb 11 7a cb
23 4d 7b 70 e0 9d 3d f0
84 b3 a8 38 70 db 8b 78
42 cc 54 1c 38 f8 d0 3c
21 66 2a 0e 1c 7c 68 1e
85 33 15 07 0e 3e 34 0f
d7 8c 9f 96 07 1f 1a 92
c3 dc 09 21 63 e2 01 97
f4 6e 91 85 a4 71 95 de
7a 37 dd d7 52 ad df 6f
3d 8e fb fe 29 c3 fa a2
8b 47 e8 7f 81 f4 7d 51
d0 b6 74 aa d5 7a ab bd
68 5b 3a 55 ff 3d c0 cb
34 b8 1d bf ea 8b 60 f0
e9 df 6d 56 96 03 06 15
e1 fa a3 2b 4b 94 03 9f
e5 7d c4 80 b0 4a 94 da
e7 ab 62 40 58 25 4a 6d
e6 c0 31 20 2c 87 25 a3
73 60 8d 10 16 d6 87 c4
ac 30 d3 08 0b 6b d6 62
56 18 fc 04 90 a0 6b 31

```

Base 6.4 vs Base 6.3

```

5a a2 68 13 38 8b d8 26
2d 51 34 9c 1c d0 6c 13
83 bd 1a 4e 0e 68 36 9c
d4 cb 0d 27 07 34 1b 4e
6a f0 93 86 96 1a 98 27
35 78 dc 43 4b 0d 4c 86
8f 3c 6e b4 b0 93 26 43
d2 1e 37 5a 58 dc 13 b4
13 3c 84 89 0c 1e 25 0e
9c 1e 42 d1 06 0f 87 07
4e 0f 21 fd 03 92 d6 96
27 92 85 eb 94 49 6b 4b
86 49 d7 e0 4a b1 a0 b0
43 b1 fe 70 25 cd 50 58
b4 cd 7f 38 87 f3 28 2c
5a f3 aa 1c d6 ec 14 16
cc 46 e2 fc 06 5a a5 d1
66 23 71 7e 03 2d c7 fd
33 84 ad 3f 94 83 f6 eb
8c 42 c3 8a 4a d4 7b e0
46 21 f4 45 25 6a a8 70
23 85 7a b7 87 35 54 38
84 d7 3d ce d6 8f 2a 1c
42 fe 8b 67 6b d2 15 0e
5c 3a 3d 1c 04 15 b6 14
2e 1d 8b 0e 02 9f 5b 0a
17 9b d0 07 01 da b8 05
9e d8 68 96 95 6d 5c 97
4f 6c 34 4b df a3 2e de
b2 36 1a b0 fa c4 17 6f
59 1b 0d 58 7d 62 9e a2
b9 98 93 2c ab 31 4f 51
23 32 d9 1d 38 dd 92 53
84 19 f9 9b 1c fb 49 bc
42 99 e9 d8 0e e8 b1 5e
21 d9 e1 6c 07 74 cd 2f
85 f9 e5 36 96 3a f3 82
d7 e9 e7 1b 4b 1d ec 41
fe e1 e6 98 b0 9b 76 b5
7f e5 73 4c 58 d8 3b cf
05 95 b0 0f 12 02 7c 6f
97 df 58 92 09 01 3e a2
de fa 2c 49 91 95 1f 51
6f 7d 16 b1 dd df 9a bd
a2 ab 0b cd fb fa 4d cb
51 c0 90 f3 e8 7d b3 f0
bd 60 48 ec 74 ab cc 78
cb 30 24 76 3a c0 66 3c
69 53 74 ee fd 12 3d e6
a1 bc 3a 77 eb 09 8b 73
c5 5e 1d ae e0 91 d0 ac
f7 2f 9b 57 70 dd 68 56
ee 82 d8 be 38 fb 34 2b
77 41 6c 5f 1c e8 1a 80
ae b5 36 ba 0e 74 0d 40
57 cf 1b 5d 07 3a 93 20
71 16 96 df 17 d7 b6 0b
ad 0b 4b fa 9e fe 5b 90
c3 90 b0 7d 4f 7f b8 48
f4 48 58 ab b2 aa 5c 24
7a 24 2c c0 59 55 2e 12
3d 12 16 60 b9 bf 17 09
8b 09 0b 30 c9 ca 9e 91
d0 91 90 18 f1 65 4f dd
c2 b3 c4 c3 2c be 19 54

```

Base 6.4 vs Base 6.3

```

61 cc 62 f4 16 5f 99 2a
a5 66 31 7a 0b ba d9 15
c7 33 8d 3d 90 5d f9 9f
f6 8c d3 8b 48 bb e9 da
7b 46 fc d0 24 c8 e1 6d
a8 23 7e 68 12 64 e5 a3
54 84 3f 34 09 32 e7 c4
53 7a 71 cc f9 5a 88 7c
bc 3d ad 66 e9 2d 44 3e
5e 8b c3 33 e1 83 22 1f
2f d0 f4 8c e5 d4 11 9a
82 68 7a 46 e7 6a 9d 4d
41 34 3d 23 e6 35 db b3
b5 1a 8b 84 73 8f f8 cc
cf 0d d0 42 ac d2 7c 66
4c 70 84 cf 5d 51 0c 52
26 38 42 f2 bb bd 06 29
13 1c 21 79 c8 cb 03 81
9c 0e 85 a9 64 f0 94 d5
4e 07 d7 c1 32 78 4a ff
27 96 fe f5 19 3c 25 ea
86 4b 7f ef 99 1e 87 75
43 b0 aa e2 d9 0f d6 af
cd f1 48 e0 31 db a8 39
f3 ed 24 70 8d f8 54 89
ec e3 12 38 d3 7c 2a d1
76 e4 09 1c fc 3e 15 fd
3b 72 91 0e 7e 1f 9f eb
88 39 dd 07 3f 9a da e0
44 89 fb 96 8a 4d 6d 70
22 d1 e8 4b 45 b3 a3 38
3e f8 df fe b6 59 13 7d
1f 7c fa 7f 5b b9 9c ab
9a 3e 7d aa b8 c9 4e c0
4d 1f ab 55 5c f1 27 60
b3 9a c0 bf 2e ed 86 30
cc 4d 60 ca 17 e3 43 18
66 b3 30 65 9e e4 b4 0c
33 cc 18 a7 4f 72 5a 06
c5 37 4b 8b 97 0f 42 d6
f7 8e b0 d0 de 92 21 6b
ee 47 58 68 6f 49 85 a0
77 b6 2c 34 a2 b1 d7 50
ae 5b 16 1a 51 cd fe 28
57 b8 0b 0d bd f3 7f 14
be 5c 90 93 cb ec aa 0a
5f 2e 48 dc f0 76 55 05
aa c6 95 0a d1 ed c2 48
55 63 df 05 fd e3 61 24
bf a4 fa 97 eb e4 a5 12
ca 52 7d de e0 72 c7 09
65 29 ab 6f 70 39 f6 91
a7 81 c0 a2 38 89 7b dd
c6 d5 60 51 1c d1 a8 fb
63 ff 30 bd 0e fd 54 e8
d1 21 3e c6 9c 2a 89 1e
fd 85 1f 63 4e 15 d1 0f
eb d7 9a a4 27 9f fd 92
e0 fe 4d 52 86 da eb 49
70 7f b3 29 43 6d e0 b1
38 aa cc 81 b4 a3 70 cd
1c 55 66 d5 5a c4 38 f3
0e bf 33 ff 2d 62 1c ec
31 42 1f 8c 75 b5 69 ad
8d 21 9a 46 af cf a1 c3

```

Base 6.4 vs Base 6.3

```

d3 85 4d 23 c2 f2 c5 f4
fc d7 b3 84 61 79 f7 7a
7e fe cc 42 a5 a9 ee 3d
3f 7f 66 21 c7 c1 77 8b
8a aa 33 85 f6 f5 ae d0
45 55 8c d7 7b ef 57 68
50 f9 b4 67 51 c7 64 d7
28 e9 5a a6 bd f6 32 fe
14 e1 2d 53 cb 7b 19 7f
0a e5 83 bc f0 a8 99 aa
05 e7 d4 5e 78 54 d9 55
97 e6 6a 2f 3c 2a f9 bf
de 73 35 82 1e 15 e9 ca
6f ac 8f 41 0f 9f e1 65
9b 0d 06 33 95 7e 70 e9
d8 93 03 8c df 3f 38 e1
6c dc 94 46 fa 8a 1c e5
36 6e 4a 23 7d 45 0e e7
1b 37 25 84 ab b7 07 e6
98 8e 87 42 c0 ce 96 73
4c 47 d6 21 60 67 4b ac
26 b6 6b 85 30 a6 b0 56
08 54 3f 90 eb 36 e1 c9
04 2a 8a 48 e0 1b e5 f1
02 15 45 24 70 98 e7 ed
01 9f b7 12 38 4c e6 e3
95 da ce 09 1c 26 73 e4
df 6d 67 91 0e 13 ac 72
fa a3 a6 dd 07 9c 56 39
7d c4 53 fb 96 4e 2b 89
06 02 b1 b9 85 bf f7 35
03 01 cd c9 d7 ca ee 8f
94 95 f3 f1 fe 65 77 d2
4a df ec ed 7f a7 ae 69
25 fa 76 e3 aa c6 57 a1
87 7d 3b e4 55 63 be c5
d6 ab 88 72 bf a4 5f f7
6b c0 44 39 ca 52 ba ee
aa ea eb 2d 15 5a b0 5d
55 75 e0 83 9f 2d 58 bb
bf af 70 d4 da 83 2c c8
ca c2 38 6a 6d d4 16 64
65 61 1c 35 a3 6a 0b 32
a7 a5 0e 8f c4 35 90 19
c6 c7 07 d2 62 8f 48 99
63 f6 96 69 31 d2 24 d9
fd 5f 19 02 2b 58 9c 1e
eb ba 99 01 80 2c 4e 0f
e0 5d d9 95 40 16 27 92
70 bb f9 df 20 0b 86 49
38 c8 e9 fa 10 90 43 b1
1c 64 e1 7d 08 48 b4 cd
0e 32 e5 ab 04 24 5a f3
07 19 e7 c0 02 12 2d ec
57 27 fb 23 e2 82 b0 c3
be 86 e8 84 71 41 58 f4
5f 43 74 42 ad b5 2c 7a
ba b4 3a 21 c3 cf 16 3d
5d 5a 1d 85 f4 f2 0b 8b
bb 2d 9b d7 7a 79 90 d0
c8 83 d8 fe 3d a9 48 68
64 d4 6c 7f 8b c1 24 34
38 71 ee fd 49 46 ac dd
1c ad 77 eb b1 23 56 fb
0e c3 ae e0 cd 84 2b e8

```

Base 6.4 vs Base 6.3

```

07 f4 57 70 f3 42 80 74
96 7a be 38 ec 21 40 3a
4b 3d 5f 1c 76 85 20 1d
b0 8b ba 0e 3b d7 10 9b
58 d0 5d 07 88 fe 08 d8
e4 5d 1a dc 22 1d 67 8b
72 bb 0d 6e 11 9b a6 d0
39 c8 93 37 9d d8 53 68
89 64 dc 8e db 6c bc 34
d1 32 6e 47 f8 36 5e 1a
fd 19 37 b6 7c 1b 2f 0d
eb 99 8e 5b 3e 98 82 93
e0 d9 47 b8 1f 4c 41 dc
76 0b b8 09 6a da 2a f5
3b 90 5c 91 35 6d 15 ef
88 48 2e dd 8f a3 9f e2
44 24 17 fb d2 c4 da 71
22 12 9e e8 69 62 6d ad
11 09 4f 74 a1 31 a3 c3
9d 91 b2 3a c5 8d c4 f4
db dd 59 1d f7 d3 62 7a
a2 ae be c3 b8 ce 2f 60
51 57 5f f4 5c 67 82 30
bd be ba 7a 2e a6 41 18
cb 5f 5d 3d 17 53 b5 0c
f0 ba bb 8b 9e bc cf 06
78 5d c8 d0 4f 5e f2 03
3c bb 64 68 b2 2f 79 94
1e c8 32 34 59 82 a9 4a
eb 37 d6 fd 0c 99 9f 2b
e0 8e 6b eb 06 d9 da 80
70 47 a0 e0 03 f9 6d 40
38 b6 50 70 94 e9 a3 20
1c 5b 28 38 4a e1 c4 10
0e b8 14 1c 25 e5 62 08
07 5c 0a 0e 87 e7 31 04
96 2e 05 07 d6 e6 8d 02
33 c9 14 08 c4 90 7c f4
8c f1 0a 04 62 48 3e 7a
46 ed 05 02 31 24 1f 3d
23 e3 97 01 8d 12 9a 8b
84 e4 de 95 d3 09 4d d0
42 72 6f df fc 91 b3 68
21 39 a2 fa 7e dd cc 34
85 89 51 7d 3f fb 66 1a
5a 22 e7 57 12 50 a7 05
2d 11 e6 be 09 28 c6 97
83 9d 73 5f 91 14 63 de
d4 db ac ba dd 0a a4 6f
6a f8 56 5d fb 05 52 a2
35 7c 2b bb e8 97 29 51
8f 3e 80 c8 74 de 81 bd
d2 1f 40 64 3a 6f d5 cb
93 b3 c0 a3 d7 61 06 0e
dc cc 60 c4 fe a5 03 07
6e 66 30 62 7f c7 94 96
37 33 18 31 aa f6 4a 4b
8e 8c 0c 8d 55 7b 25 b0
47 46 06 d3 bf a8 87 58
b6 23 03 fc ca 54 d6 2c
5b 84 94 7e 65 2a 6b 16
e4 af 8e 80 47 1e 38 fc
72 c2 47 40 b6 0f 1c 7e
39 61 b6 20 5b 92 0e 3f
89 a5 5b 10 b8 49 07 8a

```

Base 6.4 vs Base 6.3

```

d1 c7 b8 08 5c b1 96 45
fd f6 5c 04 2e cd 4b b7
eb 7b 2e 02 17 f3 b0 ce
e0 a8 17 01 9e ec 58 67
84 9f e4 6c 69 85 5d f5
42 da 72 36 a1 d7 bb ef
21 6d 39 1b c5 fe c8 e2
85 a3 89 98 f7 7f 64 71
d7 c4 d1 4c ee aa 32 ad
fe 62 fd 26 77 55 19 c3
7f 31 eb 13 ae bf 99 f4
aa 8d e0 9c 57 ca d9 7a
11 c2 af e7 77 15 77 a2
9d 61 c2 e6 ae 9f ae 51
db a5 61 73 57 da 57 bd
f8 c7 a5 ac be 6d be cb
7c f6 c7 56 5f a3 5f f0
3e 7b f6 2b ba c4 ba 78
1f a8 7b 80 5d 62 5d 3c
9a 54 a8 40 bb 31 bb 1e
b3 bd 5b 06 6b 8a 93 03
cc cb b8 03 a0 45 dc 94
66 f0 5c 94 50 b7 6e 4a
33 78 2e 4a 28 ce 37 25
8c 3c 17 25 14 67 8e 87
46 1e 9e 87 0a a6 47 d6
23 0f 4f d6 05 53 b6 6b
84 92 b2 6b 97 bc 5b a0
89 c0 af 5f 36 d8 cd 28
d1 60 c2 ba 1b 6c f3 14
fd 30 61 5d 98 36 ec 0a
eb 18 a5 bb 4c 1b 76 05
e0 0c c7 c8 26 98 3b 97
70 06 f6 64 13 4c 88 de
38 03 7b 32 9c 26 44 6f
1c 94 a8 19 4e 13 22 a2
e0 57 c6 16 66 b5 c6 71
70 be 63 0b 33 cf 63 ad
38 5f a4 90 8c f2 a4 c3
1c ba 52 48 46 79 52 f4
0e 5d 29 24 23 a9 29 7a
07 bb 81 12 84 c1 81 3d
96 c8 d5 09 42 f5 d5 8b
4b 64 ff 91 21 ef ff d0
55 5c f7 64 0e 54 cf 7a
bf 2e ee 32 07 2a f2 3d
ca 17 77 19 96 15 79 8b
65 9e ae 99 4b 9f a9 d0
a7 4f 57 d9 b0 da c1 68
c6 b2 be f9 58 6d f5 34
63 59 5f e9 2c a3 ef 1a
a4 b9 ba e1 16 c4 e2 0d
c2 ad 7e 90 f9 bb 8f 0f
61 c3 3f 48 e9 c8 d2 92
a5 f4 8a 24 e1 64 69 49
c7 7a 45 12 e5 32 a1 b1
f6 3d b7 09 e7 19 c5 cd
7b 8b ce 91 e6 99 f7 f3
a8 d0 67 dd 73 d9 ee ec
54 68 a6 fb ac f9 77 76
4d c0 22 19 fc cc d3 7c
b3 60 11 99 7e 66 fc 3e
cc 30 9d d9 3f 33 7e 1f
66 18 db f9 8a 8c 3f 9a
33 0c f8 e9 45 46 8a 4d

```

Base 6.4 vs Base 6.3

```

8c 06 7c e1 b7 23 45 b3
46 03 3e e5 ce 84 b7 cc
23 94 1f e7 67 42 ce 66
a8 3f 60 94 9f 45 cb 50
54 8a 30 4a da b7 f0 28
2a 45 18 25 6d ce 78 14
15 b7 0c 87 a3 67 3c 0a
9f ce 06 d6 c4 a6 1e 05
da 67 03 6b 62 53 0f 97
6d a6 94 a0 31 bc 92 de
a3 53 4a 50 8d 5e 49 6f
a9 04 77 09 52 a1 0b cc
c1 02 ae 91 29 c5 90 66
f5 01 57 dd 81 f7 48 33
ef 95 be fb d5 ee 24 8c
e2 df 5f e8 ff 77 12 46
71 fa ba 74 ea ae 09 23
ad 7d 5d 3a 75 57 91 84
c3 ab bb 1d af be dd 42
47 7b 14 11 85 20 da a5
b6 a8 0a 9d d7 10 6d c7
5b 54 05 db fe 08 a3 f6
b8 2a 97 f8 7f 04 c4 7b
5c 15 de 7c aa 02 62 a8
2e 9f 6f 3e 55 01 31 54
17 da a2 1f bf 95 8d 2a
9e 6d 51 9a ca df d3 15
c0 e4 9e 3e a6 90 b6 b7
60 72 4f 1f 53 48 5b ce
30 39 b2 9a bc 24 b8 67
18 89 59 4d 5e 12 5c a6
0c d1 b9 b3 2f 09 2e 53
06 fd c9 cc 82 91 17 bc
03 eb f1 66 41 dd 9e 5e
94 e0 ed 33 b5 fb 4f 2f
85 f0 5b c7 b1 77 f1 ae
d7 78 b8 f6 cd ae ed 57
fe 3c 5c 7b f3 57 e3 be
7f 1e 2e a8 ec be e4 5f
aa 0f 17 54 76 5f 72 ba
55 92 9e 2a 3b ba 39 5d
bf 49 4f 15 88 5d 89 bb
ca b1 b2 9f 44 bb d1 c8
ab 15 fa ea b6 f8 61 cb
c0 9f 7d 75 5b 7c a5 f0
60 da ab af b8 3e c7 78
30 6d c0 c2 5c 1f f6 3c
18 a3 60 61 2e 9a 7b 1e
0c c4 30 a5 17 4d a8 0f
06 62 18 c7 9e b3 54 92
03 31 0c f6 4f cc 2a 49
d7 26 20 74 ba c8 47 77
fe 13 10 3a 5d 64 b6 ae
7f 9c 08 1d bb 32 5b 57
aa 4e 04 9b c8 19 b8 be
55 27 02 d8 64 99 5c 5f
bf 86 01 6c 32 d9 2e ba
ca 43 95 36 19 f9 17 5d
65 b4 df 1b 99 e9 9e bb
9e 07 ff 02 3d d6 85 f0
4f 96 ea 01 8b 6b d7 78
b2 4b 75 95 d0 a0 fe 3c
59 b0 af df 68 50 7f 1e
b9 58 c2 fa 34 28 aa 0f
c9 2c 61 7d 1a 14 55 92

```

Base 6.4 vs Base 6.3

```

f1 16 a5 ab 0d 0a bf 49
ed 0b c7 c0 93 05 ca b1
fe 4e 75 c4 bc ad aa 2f
7f 27 af 62 5e c3 55 82
aa 86 c2 31 2f f4 bf 41
55 43 61 8d 82 7a ca b5
bf b4 a5 d3 41 3d 65 cf
ca 5a c7 fc b5 8b a7 f2
65 2d f6 7e cf d0 c6 79
a7 83 7b 3f f2 68 63 a9
12 f3 14 e0 42 77 56 78
09 ec 0a 70 21 ae 2b 3c
91 76 05 38 85 57 80 1e
dd 3b 97 1c d7 be 40 0f
fb 88 de 0e fe 5f 20 92
e8 44 6f 07 7f ba 10 49
74 22 a2 96 aa 5d 08 b1
3a 11 51 4b 55 bb 04 cd
d6 be 75 67 5c 68 9e b8
6b 5f af a6 2e 34 4f 5c
a0 ba c2 53 17 1a b2 2e
50 5d 61 bc 9e 0d 59 17
28 bb a5 5e 4f 93 b9 9e
14 c8 c7 2f b2 dc c9 4f
0a 64 f6 82 59 6e f1 b2
05 32 7b 41 b9 37 ed 59
d3 3a 12 31 ae 4e 07 99
fc 1d 09 8d 57 27 96 d9
7e 9b 91 d3 be 86 4b f9
3f d8 dd fc 5f 43 b0 e9
8a 6c fb 7e ba b4 58 e1
45 36 e8 3f 5d 5a 2c e5
b7 1b 74 8a bb 2d 16 e7
ce 98 3a 45 c8 83 0b e6
ab be 3f 3f 77 b9 74 7f
c0 5f 8a 8a ae c9 3a aa
60 ba 45 45 57 f1 1d 55
30 5d b7 b7 be ed 9b bf
18 bb ce ce 5f e3 d8 ca
0c c8 67 67 ba e4 6c 65
06 64 a6 a6 5d 72 36 a7
03 32 53 53 bb 39 1b c6
7c e3 f5 b5 fb dd f3 77
3e e4 ef cf e8 fb ec ae
1f 72 e2 f2 74 e8 76 57
9a 39 71 79 3a 74 3b be
4d 89 ad a9 1d 3a 88 5f
b3 d1 c3 c1 9b 1d 44 ba
cc fd f4 f5 d8 9b 22 5d
66 eb 7a ef 6c d8 11 bb
99 0e f4 81 6a cb 09 87
d9 07 7a d5 35 f0 91 d6
f9 96 3d ff 8f 78 dd 6b
e9 4b 8b ea d2 3c fb a0
e1 b0 d0 75 69 1e e8 50
e5 58 68 af a1 0f 74 28
e7 2c 34 c2 c5 92 3a 14
e6 16 1a 61 f7 49 1d 0a
8a 76 5a ee 38 cd 15 1b
45 3b 2d 77 1c f3 9f 98
b7 88 83 ae 0e ec da 4c
ce 44 d4 57 07 76 6d 26
67 22 6a be 96 3b a3 13
a6 11 35 5f 4b 88 c4 9c
53 9d 8f ba b0 44 62 4e

```

Base 6.4 vs Base 6.3

```

bc db d2 5d 58 22 31 27
02 1a 5e 4c 26 ae 22 ca
01 0d 2f 26 13 57 11 65
95 93 82 13 9c be 9d a7
df dc 41 9c 4e 5f db c6
fa 6e b5 4e 27 ba f8 63
7d 37 cf 27 86 5d 7c a4
ab 8e f2 86 43 bb 3e 52
c0 47 79 43 b4 c8 1f 29
9b 8e 9b a7 c8 a0 50 d2
d8 47 d8 c6 64 50 28 69
6c b6 6c 63 32 28 14 a1
36 5b 36 a4 19 14 0a c5
1b b8 1b 52 99 0a 05 f7
98 5c 98 29 d9 05 97 ee
4c 2e 4c 81 f9 97 de 77
26 17 26 d5 e9 de 6f ae
8b c9 ab cd 35 16 da c9
d0 f1 c0 f3 8f 0b 6d f1
68 ed 60 ec d2 90 a3 ed
34 e3 30 76 69 48 c4 e3
1a e4 18 3b a1 24 62 e4
0d 72 0c 88 c5 12 31 72
93 39 06 44 f7 09 8d 39
dc 89 03 22 ee 91 d3 89
68 83 83 94 ce 20 bd a3
34 d4 d4 4a 67 10 cb c4
1a 6a 6a 25 a6 08 f0 62
0d 35 35 87 53 04 78 31
93 8f 8f d6 bc 02 3c 8d
dc d2 d2 6b 5e 01 1e d3
6e 69 69 a0 2f 95 0f fc
37 a1 a1 50 82 df 92 7e
74 89 12 39 16 16 be 62
3a d1 09 89 0b 0b 5f 31
1d fd 91 d1 90 90 ba 8d
9b eb dd fd 48 48 5d d3
d8 e0 fb eb 24 24 bb fc
6c 70 e8 e0 12 12 c8 7e
36 38 74 70 09 09 64 3f
1b 1c 3a 38 91 91 32 8a
a1 d4 59 3e 12 d8 22 b2
c5 6a b9 1f 09 6c 11 59
f7 35 c9 9a 91 36 9d b9
ee 8f f1 4d dd 1b db c9
77 d2 ed b3 fb 98 f8 f1
ae 69 e3 cc e8 4c 7c ed
57 a1 e4 66 74 26 3e e3
be c5 72 33 3a 13 1f e4
c5 68 02 03 4f 57 9a 90
f7 34 01 94 b2 be 4d 48
ee 1a 95 4a 59 5f b3 24
77 0d df 25 b9 ba cc 12
ae 93 fa 87 c9 5d 66 09
57 dc 7d d6 f1 bb 33 91
be 6e ab 6b ed c8 8c dd
5f 37 c0 a0 e3 64 46 fb
f5 8f 1d d2 89 35 84 48
ef d2 9b 69 d1 8f 42 24
e2 69 d8 a1 fd d2 21 12
71 a1 6c c5 eb 69 85 09
ad c5 36 f7 e0 a1 d7 91
c3 f7 1b ee 70 c5 fe dd
f4 ee 98 77 38 f7 7f fb
7a 77 4c ae 1c ee aa e8

```

Base 6.4 vs Base 6.3

```

d5 1e 0a d3 3c c4 d8 f6
ff 0f 05 fc 1e 62 6c 7b
ea 92 97 7e 0f 31 36 a8
75 49 de 3f 92 8d 1b 54
af b1 6f 8a 49 d3 98 2a
c2 cd a2 45 b1 fc 4c 15
61 f3 51 b7 cd 7e 26 9f
a5 ec bd ce f3 3f 13 da
27 fd 8f 05 57 cb 9e 22
86 eb d2 97 be f0 4f 11
43 e0 69 de 5f 78 b2 9d
b4 70 a1 6f ba 3c 59 db
5a 38 c5 a2 5d 1e b9 f8
2d 1c f7 51 bb 0f c9 7c
83 0e ee bd c8 92 f1 3e
d4 07 77 cb 64 49 ed 1f
e3 48 2d 49 c8 21 12 e0
e4 24 83 b1 64 85 09 70
72 12 d4 cd 32 d7 91 38
39 09 6a f3 19 fe dd 1c
89 91 35 ec 99 7f fb 0e
d1 dd 8f 76 d9 aa e8 07
fd fb d2 3b f9 55 74 96
eb e8 69 88 e9 bf 3a 4b
1e 0f 81 9e cf 43 89 c1
0f 92 d5 4f f2 b4 d1 f5
92 49 ff b2 79 5a fd ef
49 b1 ea 59 a9 2d eb e2
b1 cd 75 b9 c1 83 e0 71
cd f3 af c9 f5 d4 70 ad
f3 ec c2 f1 ef 6a 38 c3
ec 76 61 ed e2 35 1c f4
51 9d af 91 17 c6 06 02
bd db c2 dd 9e 63 03 01
cb f8 61 fb 4f a4 94 95
f0 7c a5 e8 b2 52 4a df
78 3e c7 74 59 29 25 fa
3c 1f f6 3a b9 81 87 7d
1e 9a 7b 1d c9 d5 d6 ab
0f 4d a8 9b f1 ff 6b c0
2a 7e 0e a0 a7 5d e8 80
15 3f 07 50 c6 bb 74 40
9f 8a 96 28 63 c8 3a 20
da 45 4b 14 a4 64 1d 10
6d b7 b0 0a 52 32 9b 08
a3 ce 58 05 29 19 d8 04
c4 67 2c 97 81 99 6c 02
62 a6 16 de d5 d9 36 01
ce 17 d2 17 12 67 dc 33
67 9e 69 9e 09 a6 6e 8c
a6 4f a1 4f 91 53 37 46
53 b2 c5 b2 dd bc 8e 23
bc 59 f7 59 fb 5e 47 84
5e b9 ee b9 e8 2f b6 42
2f c9 77 c9 74 82 5b 21
82 f1 ae f1 3a 41 b8 85
8c da 01 89 5f ee ce c6
46 6d 95 d1 ba 77 67 63
23 a3 df fd 5d ae a6 a4
84 c4 fa eb bb 57 53 52
42 62 7d e0 c8 be bc 29
21 31 ab 70 64 5f 5e 81
85 8d c0 38 32 ba 2f d5
d7 d3 60 1c 19 5d 82 ff
06 1a 6b 83 af d8 7a 97

```

Base 6.4 vs Base 6.3

```

03 0d a0 d4 c2 6c 3d de
94 93 50 6a 61 36 8b 6f
4a dc 28 35 a5 1b d0 a2
25 6e 14 8f c7 98 68 51
87 37 0a d2 f6 4c 34 bd
d6 8e 05 69 7b 26 1a cb
6b 47 97 a1 a8 13 0d f0
b2 30 d1 07 72 d7 12 5d
59 18 fd 96 39 fe 09 bb
b9 0c eb 4b 89 7f 91 c8
c9 06 e0 b0 d1 aa dd 64
f1 03 70 58 fd 55 fb 32
ed 94 38 2c eb bf e8 19
e3 4a 1c 16 e0 ca 74 99
e4 25 0e 0b 70 65 3a d9
d1 22 50 93 58 18 de 41
fd 11 28 dc 2c 0c 6f b5
eb 9d 14 6e 16 06 a2 cf
e0 db 0a 37 0b 03 51 f2
70 f8 05 8e 90 94 bd 79
38 7c 97 47 48 4a cb a9
1c 3e de b6 24 25 f0 c1
0e 1f 6f 5b 12 87 78 f5
32 2c 4a 48 47 e2 36 ad
19 16 25 24 b6 71 1b c3
99 0b 87 12 5b ad 98 f4
d9 90 d6 09 b8 c3 4c 7a
f9 48 6b 91 5c f4 26 3d
e9 24 a0 dd 2e 7a 13 8b
e1 12 50 fb 17 3d 9c d0
e5 09 28 e8 9e 8b 4e 68
6a 14 59 01 53 5b b3 6c
35 0a b9 95 bc b8 cc 36
8f 05 c9 df 5e 5c 66 1b
d2 97 f1 fa 2f 2e 33 98
69 de ed 7d 82 17 8c 4c
a1 6f e3 ab 41 9e 46 26
c5 a2 e4 c0 b5 4f 23 13
f7 51 72 60 cf b2 84 9c
62 83 50 25 0b 9a eb b0
31 d4 28 87 90 4d e0 58
8d 6a 14 d6 48 b3 70 2c
d3 35 0a 6b 24 cc 38 16
fc 8f 05 a0 12 66 1c 0b
7e d2 97 50 09 33 0e 90
3f 69 de 28 91 8c 07 48
8a a1 6f 14 dd 46 96 24
a7 b7 55 2f 79 9c 09 85
c6 ce bf 82 a9 4e 91 d7
63 67 ca 41 c1 27 dd fe
a4 a6 65 b5 f5 86 fb 7f
52 53 a7 cf ef 43 e8 aa
29 bc c6 f2 e2 b4 74 55
81 5e 63 79 71 5a 3a bf
d5 2f a4 a9 ad 2d 1d ca
0e 3f 41 c0 a1 d1 28 cd
07 8a b5 60 c5 fd 14 f3
96 45 cf 30 f7 eb 0a ec
4b b7 f2 18 ee e0 05 76
b0 ce 79 0c 77 70 97 3b
58 67 a9 06 ae 38 de 88
2c a6 c1 03 57 1c 6f 44
16 53 f5 94 be 0e a2 22
c5 27 b3 5b c8 db 76 68
f7 86 cc b8 64 f8 3b 34

```

Base 6.4 vs Base 6.3

```

ee 43 66 5c 32 7c 88 1a
77 b4 33 2e 19 3e 44 0d
ae 5a 8c 17 99 1f 22 93
57 2d 46 9e d9 9a 11 dc
be 83 23 4f f9 4d 9d 6e
5f d4 84 b2 e9 b3 db 37
ba 3e 45 55 05 d9 7c 48
5d 1f b7 bf 97 f9 3e 24
bb 9a ce ca de e9 1f 12
c8 4d 67 65 6f e1 9a 09
64 b3 a6 a7 a2 e5 4d 91
32 cc 53 c6 51 e7 b3 dd
19 66 bc 63 bd e6 cc fb
99 33 5e a4 cb 73 66 e8
8d 8b 23 b6 e5 28 f5 74
d3 d0 84 5b e7 14 ef 3a
fc 68 42 b8 e6 0a e2 1d
7e 34 21 5c 73 05 71 9b
3f 1a 85 2e ac 97 ad d8
8a 0d d7 17 56 de c3 6c
45 93 fe 9e 2b 6f f4 36
b7 dc 7f 4f 80 a2 7a 1b
82 50 aa ec 5a a2 85 fe
41 28 55 76 2d 51 d7 7f
b5 14 bf 3b 83 bd fe aa
cf 0a ca 88 d4 cb 7f 55
f2 05 65 44 6a f0 aa bf
79 97 a7 22 35 78 55 ca
a9 de c6 11 8f 3c bf 65
c1 6f 63 9d d2 1e ca a7
76 d7 7d 7c fa 60 f9 ff
3b fe ab 3e 7d 30 e9 ea
88 7f c0 1f ab 18 e1 75
44 aa 60 9a c0 0c e5 af
22 55 30 4d 60 06 e7 c2
11 bf 18 b3 30 03 e6 61
9d ca 0c cc 18 94 73 a5
db 65 06 66 0c 4a ac c7
7e 6b cb 53 02 1d 25 60
3f a0 f0 bc 01 9b 87 30
8a 50 78 5e 95 d8 d6 18
45 28 3c 2f df 6c 6b 0c
b7 14 1e 82 fa 36 a0 06
ce 0a 0f 41 7d 1b 50 03
67 05 92 b5 ab 98 28 94
a6 97 49 cf c0 4c 14 4a
90 e0 c5 f9 cc 9f 84 55
48 70 f7 e9 66 da 42 bf
24 38 ee e1 33 6d 21 ca
12 1c 77 e5 8c a3 85 65
09 0e ae e7 46 c4 d7 a7
91 07 57 e6 23 62 fe c6
dd 96 be 73 84 31 7f 63
fb 4b 5f ac 42 8d aa a4
e3 12 fd cf ec 5f b4 47
e4 09 eb f2 76 ba 5a b6
72 91 e0 79 3b 5d 2d 5b
39 dd 70 a9 88 bb 83 b8
89 fb 38 c1 44 c8 d4 5c
d1 e8 1c f5 22 64 6a 2e
fd 74 0e ef 11 32 35 17
eb 3a 07 e2 9d 19 8f 9e
44 df 07 ba b1 e5 2e c1
22 fa 96 5d cd e7 17 f5
11 7d 4b bb f3 e6 9e ef

```

Base 6.4 vs Base 6.3

```

9d ab b0 c8 ec 73 4f e2
db c0 58 64 76 ac b2 71
f8 60 2c 32 3b 56 59 ad
7c 30 16 19 88 2b b9 c3
3e 18 0b 99 44 80 c9 f4
30 4f 1c 5e 36 a7 05 0c
18 b2 0e 2f 1b c6 97 06
0c 59 07 82 98 63 de 03
06 b9 96 41 4c a4 6f 94
03 c9 4b b5 26 52 a2 4a
94 f1 b0 cf 13 29 51 25
4a ed 58 f2 9c 81 bd 87
25 e3 2c 79 4e d5 cb d6
88 92 98 f1 70 ea 88 be
44 49 4c ed 38 75 44 5f
22 b1 26 e3 1c af 22 ba
11 cd 13 e4 0e c2 11 5d
9d f3 9c 72 07 61 9d bb
db ec 4e 39 96 a5 db c8
f8 76 27 89 4b c7 f8 64
7c 3b 86 d1 b0 f6 7c 32
67 08 96 85 67 b1 1d 18
a6 04 4b d7 a6 cd 9b 0c
53 02 b0 fe 53 f3 d8 06
bc 01 58 7f bc ec 6c 03
5e 95 2c aa 5e 76 36 94
2f df 16 55 2f 3b 1b 4a
82 fa 0b bf 82 88 98 25
41 7d 90 ca 41 44 4c 87
d0 92 8e bf ad 04 f3 63
68 49 47 ca c3 02 ec a4
34 b1 b6 65 f4 01 76 52
1a cd 5b a7 7a 95 3b 29
0d f3 b8 c6 3d df 88 81
93 ec 5c 63 8b fa 44 d5
dc 76 2e a4 d0 7d 22 ff
6e 3b 17 52 68 ab 11 ea
57 9a 98 68 68 8e 59 c4
be 4d 4c 34 34 47 b9 62
5f b3 26 1a 1a b6 c9 31
ba cc 13 0d 0d 5b f1 8d
5d 66 9c 93 93 b8 ed d3
bb 33 4e dc dc 5c e3 fc
c8 8c 27 6e 6e 2e e4 7e
64 46 86 37 37 17 72 3f
85 12 a5 77 45 af ab dd
d7 09 c7 ae b7 c2 c0 fb
fe 91 f6 57 ce 61 60 e8
7f dd 7b be 67 a5 30 74
aa fb a8 5f a6 c7 18 3a
55 e8 54 ba 53 f6 0c 1d
bf 74 2a 5d bc 7b 06 9b
ca 3a 15 bb 5e a8 03 d8
49 eb 4a 1e 6e a2 12 cb
b1 e0 25 0f 37 51 09 f0
cd 70 87 92 8e bd 91 78
f3 38 d6 49 47 cb dd 3c
ec 1c 6b b1 b6 f0 fb 1e
76 0e a0 cd 5b 78 e8 0f
3b 07 50 f3 b8 3c 74 92
88 96 28 ec 5c 1e 3a 49
aa dc e2 2f 3d ab 63 df
55 6e 71 82 8b c0 a4 fa
bf 37 ad 41 d0 60 52 7d
ca 8e c3 b5 68 30 29 ab

```

Base 6.4 vs Base 6.3

```
65 47 f4 cf 34 18 81 c0
a7 b6 7a f2 1a 0c d5 60
c6 5b 3d 79 0d 06 ff 30
63 b8 8b a9 93 03 ea 18
cb 56 1b 2a da 8b 1e 1e
f0 2b 98 15 6d d0 0f 0f
78 80 4c 9f a3 68 92 92
3c 40 26 da c4 34 49 49
1e 20 13 6d 62 1a b1 b1
0f 10 9c a3 31 0d cd cd
92 08 4e c4 8d 93 f3 f3
49 04 27 62 d3 dc ec ec
31 f0 47 bd 4c 42 b1 20
8d 78 b6 cb 26 21 cd 10
d3 3c 5b f0 13 85 f3 08
fc 1e b8 78 9c d7 ec 04
7e 0f 5c 3c 4e fe 76 02
3f 92 2e 1e 27 7f 3b 01
8a 49 17 0f 86 aa 88 95
45 b1 9e 92 43 55 44 df
e2 a1 85 5e a6 1f e9 d7
71 c5 d7 2f 53 9a e1 fe
ad f7 fe 82 bc 4d e5 7f
c3 ee 7f 41 5e b3 e7 aa
f4 77 aa b5 2f cc e6 55
7a ae 55 cf 82 66 73 bf
3d 57 bf f2 41 33 ac ca
8b be ca 79 b5 8c 56 65
22 cf 68 25 c2 f9 f3 a8
11 f2 34 87 61 e9 ec 54
9d 79 1a d6 a5 e1 76 2a
db a9 0d 6b c7 e5 3b 15
f8 c1 93 a0 f6 e7 88 9f
7c f5 dc 50 7b e6 44 da
3e ef 6e 28 a8 73 22 6d
1f e2 37 14 54 ac 11 a3
2d 4c 76 20 05 22 ca 06
83 26 3b 10 97 11 65 03
d4 13 88 08 de 9d a7 94
6a 9c 44 04 6f db c6 4a
35 4e 22 02 a2 f8 63 25
8f 27 11 01 51 7c a4 87
d2 86 9d 95 bd 3e 52 d6
69 43 db df cb 1f 29 6b
81 5c fe c8 f4 a9 92 07
d5 2e 7f 64 7a c1 49 96
ff 17 aa 32 3d f5 b1 4b
ea 9e 55 19 8b ef cd b0
75 4f bf 99 d0 e2 f3 58
af b2 ca d9 68 71 ec 2c
c2 59 65 f9 34 ad 76 16
61 b9 a7 e9 1a c3 3b 0b
```

Base 6.4 vs Base 6.3

K.2 Flit 8 byte LCRC §

pci_sig_8B_crc.sv

Base 6.4 vs Base 6.3

```

`ifndef PCI_SIG_8B_CRC__SV
`define PCI_SIG_8B_CRC__SV
//PCI SIG code for Flit Mode 8B CRC generation
//input: 242 bytes of flit data
//output: 250 bytes of flit data + CRC
module pci_sig_8B_crc (
    input logic [7:0] data_in[241:0],
    output logic [7:0] data_out[249:0]
);
always_comb begin
    //Assign data bytes from input to output
    for (int i =0;i<242;i++) begin
        data_out[i] = data_in[i];
    end
    //Assign CRC bytes
    data_out[249] = ({8{data_in[0][0]}} & 8'h61)^
                    ({8{data_in[0][1]}} & 8'hc2)^
                    ({8{data_in[0][2]}} & 8'haf)^
                    ({8{data_in[0][3]}} & 8'h75)^
                    ({8{data_in[0][4]}} & 8'hea)^
                    ({8{data_in[0][5]}} & 8'hff)^
                    ({8{data_in[0][6]}} & 8'hd5)^
                    ({8{data_in[0][7]}} & 8'h81)^
                    ({8{data_in[1][0]}} & 8'h69)^
                    ({8{data_in[1][1]}} & 8'hd2)^
                    ({8{data_in[1][2]}} & 8'h8f)^
                    ({8{data_in[1][3]}} & 8'h35)^
                    ({8{data_in[1][4]}} & 8'h6a)^
                    ({8{data_in[1][5]}} & 8'hd4)^
                    ({8{data_in[1][6]}} & 8'h83)^
                    ({8{data_in[1][7]}} & 8'h2d)^
                    ({8{data_in[2][0]}} & 8'h1f)^
                    ({8{data_in[2][1]}} & 8'h3e)^
                    ({8{data_in[2][2]}} & 8'h7c)^
                    ({8{data_in[2][3]}} & 8'hf8)^
                    ({8{data_in[2][4]}} & 8'hdb)^
                    ({8{data_in[2][5]}} & 8'h9d)^
                    ({8{data_in[2][6]}} & 8'h11)^
                    ({8{data_in[2][7]}} & 8'h22)^
                    ({8{data_in[3][0]}} & 8'h8b)^
                    ({8{data_in[3][1]}} & 8'h3d)^
                    ({8{data_in[3][2]}} & 8'h7a)^
                    ({8{data_in[3][3]}} & 8'hf4)^
                    ({8{data_in[3][4]}} & 8'hc3)^
                    ({8{data_in[3][5]}} & 8'had)^
                    ({8{data_in[3][6]}} & 8'h71)^
                    ({8{data_in[3][7]}} & 8'he2)^
                    ({8{data_in[4][0]}} & 8'h45)^
                    ({8{data_in[4][1]}} & 8'h8a)^
                    ({8{data_in[4][2]}} & 8'h3f)^
                    ({8{data_in[4][3]}} & 8'h7e)^
                    ({8{data_in[4][4]}} & 8'hfc)^
                    ({8{data_in[4][5]}} & 8'hd3)^
                    ({8{data_in[4][6]}} & 8'h8d)^
                    ({8{data_in[4][7]}} & 8'h31)^
                    ({8{data_in[5][0]}} & 8'h49)^
                    ({8{data_in[5][1]}} & 8'h92)^
                    ({8{data_in[5][2]}} & 8'hf)^
                    ({8{data_in[5][3]}} & 8'h1e)^
                    ({8{data_in[5][4]}} & 8'h3c)^
                    ({8{data_in[5][5]}} & 8'h78)^
                    ({8{data_in[5][6]}} & 8'hf0)^
                    ({8{data_in[5][7]}} & 8'hcb)^
                    ({8{data_in[6][0]}} & 8'h63)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[6][1]}} & 8'hc6)^
({{8{data_in[6][2]}} & 8'ha7)^
({{8{data_in[6][3]}} & 8'h65)^
({{8{data_in[6][4]}} & 8'hca)^
({{8{data_in[6][5]}} & 8'hbf)^
({{8{data_in[6][6]}} & 8'h55)^
({{8{data_in[6][7]}} & 8'haa)^
({{8{data_in[7][0]}} & 8'h88)^
({{8{data_in[7][1]}} & 8'h3b)^
({{8{data_in[7][2]}} & 8'h76)^
({{8{data_in[7][3]}} & 8'hec)^
({{8{data_in[7][4]}} & 8'hf3)^
({{8{data_in[7][5]}} & 8'hcd)^
({{8{data_in[7][6]}} & 8'hb1)^
({{8{data_in[7][7]}} & 8'h49)^
({{8{data_in[8][0]}} & 8'hca)^
({{8{data_in[8][1]}} & 8'hbf)^
({{8{data_in[8][2]}} & 8'h55)^
({{8{data_in[8][3]}} & 8'haa)^
({{8{data_in[8][4]}} & 8'h7f)^
({{8{data_in[8][5]}} & 8'hfe)^
({{8{data_in[8][6]}} & 8'hd7)^
({{8{data_in[8][7]}} & 8'h85)^
({{8{data_in[9][0]}} & 8'h64)^
({{8{data_in[9][1]}} & 8'hc8)^
({{8{data_in[9][2]}} & 8'hbb)^
({{8{data_in[9][3]}} & 8'h5d)^
({{8{data_in[9][4]}} & 8'hba)^
({{8{data_in[9][5]}} & 8'h5f)^
({{8{data_in[9][6]}} & 8'hbe)^
({{8{data_in[9][7]}} & 8'h57)^
({{8{data_in[10][0]}} & 8'h6e)^
({{8{data_in[10][1]}} & 8'hdc)^
({{8{data_in[10][2]}} & 8'h93)^
({{8{data_in[10][3]}} & 8'hd)^
({{8{data_in[10][4]}} & 8'h1a)^
({{8{data_in[10][5]}} & 8'h34)^
({{8{data_in[10][6]}} & 8'h68)^
({{8{data_in[10][7]}} & 8'hd0)^
({{8{data_in[11][0]}} & 8'h41)^
({{8{data_in[11][1]}} & 8'h82)^
({{8{data_in[11][2]}} & 8'h2f)^
({{8{data_in[11][3]}} & 8'h5e)^
({{8{data_in[11][4]}} & 8'hbc)^
({{8{data_in[11][5]}} & 8'h53)^
({{8{data_in[11][6]}} & 8'ha6)^
({{8{data_in[11][7]}} & 8'h67)^
({{8{data_in[12][0]}} & 8'h7c)^
({{8{data_in[12][1]}} & 8'hf8)^
({{8{data_in[12][2]}} & 8'hdb)^
({{8{data_in[12][3]}} & 8'h9d)^
({{8{data_in[12][4]}} & 8'h11)^
({{8{data_in[12][5]}} & 8'h22)^
({{8{data_in[12][6]}} & 8'h44)^
({{8{data_in[12][7]}} & 8'h88)^
({{8{data_in[13][0]}} & 8'h25)^
({{8{data_in[13][1]}} & 8'h4a)^
({{8{data_in[13][2]}} & 8'h94)^
({{8{data_in[13][3]}} & 8'h3)^
({{8{data_in[13][4]}} & 8'h6)^
({{8{data_in[13][5]}} & 8'hc)^
({{8{data_in[13][6]}} & 8'h18)^
({{8{data_in[13][7]}} & 8'h30)^
({{8{data_in[14][0]}} & 8'h3e)^
({{8{data_in[14][1]}} & 8'h7c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[14][2]}} & 8'hf8)^
({{8{data_in[14][3]}} & 8'hdb)^
({{8{data_in[14][4]}} & 8'h9d)^
({{8{data_in[14][5]}} & 8'h11)^
({{8{data_in[14][6]}} & 8'h22)^
({{8{data_in[14][7]}} & 8'h44)^
({{8{data_in[15][0]}} & 8'heb)^
({{8{data_in[15][1]}} & 8'hfd)^
({{8{data_in[15][2]}} & 8'hd1)^
({{8{data_in[15][3]}} & 8'h89)^
({{8{data_in[15][4]}} & 8'h39)^
({{8{data_in[15][5]}} & 8'h72)^
({{8{data_in[15][6]}} & 8'he4)^
({{8{data_in[15][7]}} & 8'he3)^
({{8{data_in[16][0]}} & 8'hfb)^
({{8{data_in[16][1]}} & 8'hdd)^
({{8{data_in[16][2]}} & 8'h91)^
({{8{data_in[16][3]}} & 8'h9)^
({{8{data_in[16][4]}} & 8'h12)^
({{8{data_in[16][5]}} & 8'h24)^
({{8{data_in[16][6]}} & 8'h48)^
({{8{data_in[16][7]}} & 8'h90)^
({{8{data_in[17][0]}} & 8'ha6)^
({{8{data_in[17][1]}} & 8'h67)^
({{8{data_in[17][2]}} & 8'hce)^
({{8{data_in[17][3]}} & 8'hb7)^
({{8{data_in[17][4]}} & 8'h45)^
({{8{data_in[17][5]}} & 8'h8a)^
({{8{data_in[17][6]}} & 8'h3f)^
({{8{data_in[17][7]}} & 8'h7e)^
({{8{data_in[18][0]}} & 8'hdb)^
({{8{data_in[18][1]}} & 8'h9d)^
({{8{data_in[18][2]}} & 8'h11)^
({{8{data_in[18][3]}} & 8'h22)^
({{8{data_in[18][4]}} & 8'h44)^
({{8{data_in[18][5]}} & 8'h88)^
({{8{data_in[18][6]}} & 8'h3b)^
({{8{data_in[18][7]}} & 8'h76)^
({{8{data_in[19][0]}} & 8'hc1)^
({{8{data_in[19][1]}} & 8'ha9)^
({{8{data_in[19][2]}} & 8'h79)^
({{8{data_in[19][3]}} & 8'hf2)^
({{8{data_in[19][4]}} & 8'hcf)^
({{8{data_in[19][5]}} & 8'hb5)^
({{8{data_in[19][6]}} & 8'h41)^
({{8{data_in[19][7]}} & 8'h82)^
({{8{data_in[20][0]}} & 8'hb7)^
({{8{data_in[20][1]}} & 8'h45)^
({{8{data_in[20][2]}} & 8'h8a)^
({{8{data_in[20][3]}} & 8'h3f)^
({{8{data_in[20][4]}} & 8'h7e)^
({{8{data_in[20][5]}} & 8'hfc)^
({{8{data_in[20][6]}} & 8'hd3)^
({{8{data_in[20][7]}} & 8'h8d)^
({{8{data_in[21][0]}} & 8'h99)^
({{8{data_in[21][1]}} & 8'h19)^
({{8{data_in[21][2]}} & 8'h32)^
({{8{data_in[21][3]}} & 8'h64)^
({{8{data_in[21][4]}} & 8'hc8)^
({{8{data_in[21][5]}} & 8'hbb)^
({{8{data_in[21][6]}} & 8'h5d)^
({{8{data_in[21][7]}} & 8'hba)^
({{8{data_in[22][0]}} & 8'h5f)^
({{8{data_in[22][1]}} & 8'hbe)^
({{8{data_in[22][2]}} & 8'h57})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[22][3]}} & 8'hae)^
({{8{data_in[22][4]}} & 8'h77)^
({{8{data_in[22][5]}} & 8'hee)^
({{8{data_in[22][6]}} & 8'hf7)^
({{8{data_in[22][7]}} & 8'hc5)^
({{8{data_in[23][0]}} & 8'h16)^
({{8{data_in[23][1]}} & 8'h2c)^
({{8{data_in[23][2]}} & 8'h58)^
({{8{data_in[23][3]}} & 8'hb0)^
({{8{data_in[23][4]}} & 8'h4b)^
({{8{data_in[23][5]}} & 8'h96)^
({{8{data_in[23][6]}} & 8'h7)^
({{8{data_in[23][7]}} & 8'he)^
({{8{data_in[24][0]}} & 8'hd5)^
({{8{data_in[24][1]}} & 8'h81)^
({{8{data_in[24][2]}} & 8'h29)^
({{8{data_in[24][3]}} & 8'h52)^
({{8{data_in[24][4]}} & 8'ha4)^
({{8{data_in[24][5]}} & 8'h63)^
({{8{data_in[24][6]}} & 8'hc6)^
({{8{data_in[24][7]}} & 8'ha7)^
({{8{data_in[25][0]}} & 8'h8a)^
({{8{data_in[25][1]}} & 8'h3f)^
({{8{data_in[25][2]}} & 8'h7e)^
({{8{data_in[25][3]}} & 8'hfc)^
({{8{data_in[25][4]}} & 8'hd3)^
({{8{data_in[25][5]}} & 8'h8d)^
({{8{data_in[25][6]}} & 8'h31)^
({{8{data_in[25][7]}} & 8'h62)^
({{8{data_in[26][0]}} & 8'hf7)^
({{8{data_in[26][1]}} & 8'hc5)^
({{8{data_in[26][2]}} & 8'ha1)^
({{8{data_in[26][3]}} & 8'h69)^
({{8{data_in[26][4]}} & 8'hd2)^
({{8{data_in[26][5]}} & 8'h8f)^
({{8{data_in[26][6]}} & 8'h35)^
({{8{data_in[26][7]}} & 8'h6a)^
({{8{data_in[27][0]}} & 8'he5)^
({{8{data_in[27][1]}} & 8'he1)^
({{8{data_in[27][2]}} & 8'he9)^
({{8{data_in[27][3]}} & 8'hf9)^
({{8{data_in[27][4]}} & 8'hd9)^
({{8{data_in[27][5]}} & 8'h99)^
({{8{data_in[27][6]}} & 8'h19)^
({{8{data_in[27][7]}} & 8'h32)^
({{8{data_in[28][0]}} & 8'he)^
({{8{data_in[28][1]}} & 8'h1c)^
({{8{data_in[28][2]}} & 8'h38)^
({{8{data_in[28][3]}} & 8'h70)^
({{8{data_in[28][4]}} & 8'he0)^
({{8{data_in[28][5]}} & 8'heb)^
({{8{data_in[28][6]}} & 8'hfd)^
({{8{data_in[28][7]}} & 8'hd1)^
({{8{data_in[29][0]}} & 8'he4)^
({{8{data_in[29][1]}} & 8'he3)^
({{8{data_in[29][2]}} & 8'hed)^
({{8{data_in[29][3]}} & 8'hf1)^
({{8{data_in[29][4]}} & 8'hc9)^
({{8{data_in[29][5]}} & 8'hb9)^
({{8{data_in[29][6]}} & 8'h59)^
({{8{data_in[29][7]}} & 8'hb2)^
({{8{data_in[30][0]}} & 8'h6b)^
({{8{data_in[30][1]}} & 8'hd6)^
({{8{data_in[30][2]}} & 8'h87)^
({{8{data_in[30][3]}} & 8'h25)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[30][4]}} & 8'h4a)^
({{8{data_in[30][5]}} & 8'h94)^
({{8{data_in[30][6]}} & 8'h3)^
({{8{data_in[30][7]}} & 8'h6)^
({{8{data_in[31][0]}} & 8'hd7)^
({{8{data_in[31][1]}} & 8'h85)^
({{8{data_in[31][2]}} & 8'h21)^
({{8{data_in[31][3]}} & 8'h42)^
({{8{data_in[31][4]}} & 8'h84)^
({{8{data_in[31][5]}} & 8'h23)^
({{8{data_in[31][6]}} & 8'h46)^
({{8{data_in[31][7]}} & 8'h8c)^
({{8{data_in[32][0]}} & 8'h82)^
({{8{data_in[32][1]}} & 8'h2f)^
({{8{data_in[32][2]}} & 8'h5e)^
({{8{data_in[32][3]}} & 8'hbc)^
({{8{data_in[32][4]}} & 8'h53)^
({{8{data_in[32][5]}} & 8'ha6)^
({{8{data_in[32][6]}} & 8'h67)^
({{8{data_in[32][7]}} & 8'hce)^
({{8{data_in[33][0]}} & 8'h62)^
({{8{data_in[33][1]}} & 8'hc4)^
({{8{data_in[33][2]}} & 8'ha3)^
({{8{data_in[33][3]}} & 8'h6d)^
({{8{data_in[33][4]}} & 8'hda)^
({{8{data_in[33][5]}} & 8'h9f)^
({{8{data_in[33][6]}} & 8'h15)^
({{8{data_in[33][7]}} & 8'h2a)^
({{8{data_in[34][0]}} & 8'hf)^
({{8{data_in[34][1]}} & 8'h1e)^
({{8{data_in[34][2]}} & 8'h3c)^
({{8{data_in[34][3]}} & 8'h78)^
({{8{data_in[34][4]}} & 8'hf0)^
({{8{data_in[34][5]}} & 8'hcb)^
({{8{data_in[34][6]}} & 8'hbd)^
({{8{data_in[34][7]}} & 8'h51)^
({{8{data_in[35][0]}} & 8'hec)^
({{8{data_in[35][1]}} & 8'hf3)^
({{8{data_in[35][2]}} & 8'hcd)^
({{8{data_in[35][3]}} & 8'hb1)^
({{8{data_in[35][4]}} & 8'h49)^
({{8{data_in[35][5]}} & 8'h92)^
({{8{data_in[35][6]}} & 8'hf)^
({{8{data_in[35][7]}} & 8'h1e)^
({{8{data_in[36][0]}} & 8'heb)^
({{8{data_in[36][1]}} & 8'hfd)^
({{8{data_in[36][2]}} & 8'hd1)^
({{8{data_in[36][3]}} & 8'h89)^
({{8{data_in[36][4]}} & 8'h39)^
({{8{data_in[36][5]}} & 8'h72)^
({{8{data_in[36][6]}} & 8'he4)^
({{8{data_in[36][7]}} & 8'he3)^
({{8{data_in[37][0]}} & 8'hd4)^
({{8{data_in[37][1]}} & 8'h83)^
({{8{data_in[37][2]}} & 8'h2d)^
({{8{data_in[37][3]}} & 8'h5a)^
({{8{data_in[37][4]}} & 8'hb4)^
({{8{data_in[37][5]}} & 8'h43)^
({{8{data_in[37][6]}} & 8'h86)^
({{8{data_in[37][7]}} & 8'h27)^
({{8{data_in[38][0]}} & 8'ha5)^
({{8{data_in[38][1]}} & 8'h61)^
({{8{data_in[38][2]}} & 8'hc2)^
({{8{data_in[38][3]}} & 8'haf)^
({{8{data_in[38][4]}} & 8'h75})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[38][5]}} & 8'hea)^
({{8{data_in[38][6]}} & 8'hff)^
({{8{data_in[38][7]}} & 8'hd5)^
({{8{data_in[39][0]}} & 8'h7a)^
({{8{data_in[39][1]}} & 8'hf4)^
({{8{data_in[39][2]}} & 8'hc3)^
({{8{data_in[39][3]}} & 8'had)^
({{8{data_in[39][4]}} & 8'h71)^
({{8{data_in[39][5]}} & 8'he2)^
({{8{data_in[39][6]}} & 8'hef)^
({{8{data_in[39][7]}} & 8'hf5)^
({{8{data_in[40][0]}} & 8'h5f)^
({{8{data_in[40][1]}} & 8'hbe)^
({{8{data_in[40][2]}} & 8'h57)^
({{8{data_in[40][3]}} & 8'hae)^
({{8{data_in[40][4]}} & 8'h77)^
({{8{data_in[40][5]}} & 8'hee)^
({{8{data_in[40][6]}} & 8'hf7)^
({{8{data_in[40][7]}} & 8'hc5)^
({{8{data_in[41][0]}} & 8'hbe)^
({{8{data_in[41][1]}} & 8'h57)^
({{8{data_in[41][2]}} & 8'hae)^
({{8{data_in[41][3]}} & 8'h77)^
({{8{data_in[41][4]}} & 8'hee)^
({{8{data_in[41][5]}} & 8'hf7)^
({{8{data_in[41][6]}} & 8'hc5)^
({{8{data_in[41][7]}} & 8'ha1)^
({{8{data_in[42][0]}} & 8'h1b)^
({{8{data_in[42][1]}} & 8'h36)^
({{8{data_in[42][2]}} & 8'h6c)^
({{8{data_in[42][3]}} & 8'hd8)^
({{8{data_in[42][4]}} & 8'h9b)^
({{8{data_in[42][5]}} & 8'h1d)^
({{8{data_in[42][6]}} & 8'h3a)^
({{8{data_in[42][7]}} & 8'h74)^
({{8{data_in[43][0]}} & 8'h37)^
({{8{data_in[43][1]}} & 8'h6e)^
({{8{data_in[43][2]}} & 8'hdc)^
({{8{data_in[43][3]}} & 8'h93)^
({{8{data_in[43][4]}} & 8'hd)^
({{8{data_in[43][5]}} & 8'h1a)^
({{8{data_in[43][6]}} & 8'h34)^
({{8{data_in[43][7]}} & 8'h68)^
({{8{data_in[44][0]}} & 8'hdc)^
({{8{data_in[44][1]}} & 8'h93)^
({{8{data_in[44][2]}} & 8'hd)^
({{8{data_in[44][3]}} & 8'h1a)^
({{8{data_in[44][4]}} & 8'h34)^
({{8{data_in[44][5]}} & 8'h68)^
({{8{data_in[44][6]}} & 8'hd0)^
({{8{data_in[44][7]}} & 8'h8b)^
({{8{data_in[45][0]}} & 8'h26)^
({{8{data_in[45][1]}} & 8'h4c)^
({{8{data_in[45][2]}} & 8'h98)^
({{8{data_in[45][3]}} & 8'h1b)^
({{8{data_in[45][4]}} & 8'h36)^
({{8{data_in[45][5]}} & 8'h6c)^
({{8{data_in[45][6]}} & 8'hd8)^
({{8{data_in[45][7]}} & 8'h9b)^
({{8{data_in[46][0]}} & 8'hc0)^
({{8{data_in[46][1]}} & 8'hab)^
({{8{data_in[46][2]}} & 8'h7d)^
({{8{data_in[46][3]}} & 8'hfa)^
({{8{data_in[46][4]}} & 8'hdf)^
({{8{data_in[46][5]}} & 8'h95)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[46][6]}} & 8'h1)^
({{8{data_in[46][7]}} & 8'h2)^
({{8{data_in[47][0]}} & 8'hbc)^
({{8{data_in[47][1]}} & 8'h53)^
({{8{data_in[47][2]}} & 8'ha6)^
({{8{data_in[47][3]}} & 8'h67)^
({{8{data_in[47][4]}} & 8'hce)^
({{8{data_in[47][5]}} & 8'hb7)^
({{8{data_in[47][6]}} & 8'h45)^
({{8{data_in[47][7]}} & 8'h8a)^
({{8{data_in[48][0]}} & 8'he6)^
({{8{data_in[48][1]}} & 8'he7)^
({{8{data_in[48][2]}} & 8'he5)^
({{8{data_in[48][3]}} & 8'he1)^
({{8{data_in[48][4]}} & 8'he9)^
({{8{data_in[48][5]}} & 8'hf9)^
({{8{data_in[48][6]}} & 8'hd9)^
({{8{data_in[48][7]}} & 8'h99)^
({{8{data_in[49][0]}} & 8'h66)^
({{8{data_in[49][1]}} & 8'hcc)^
({{8{data_in[49][2]}} & 8'hb3)^
({{8{data_in[49][3]}} & 8'h4d)^
({{8{data_in[49][4]}} & 8'h9a)^
({{8{data_in[49][5]}} & 8'h1f)^
({{8{data_in[49][6]}} & 8'h3e)^
({{8{data_in[49][7]}} & 8'h7c)^
({{8{data_in[50][0]}} & 8'h3)^
({{8{data_in[50][1]}} & 8'h6)^
({{8{data_in[50][2]}} & 8'hc)^
({{8{data_in[50][3]}} & 8'h18)^
({{8{data_in[50][4]}} & 8'h30)^
({{8{data_in[50][5]}} & 8'h60)^
({{8{data_in[50][6]}} & 8'hc0)^
({{8{data_in[50][7]}} & 8'hab)^
({{8{data_in[51][0]}} & 8'hce)^
({{8{data_in[51][1]}} & 8'hb7)^
({{8{data_in[51][2]}} & 8'h45)^
({{8{data_in[51][3]}} & 8'h8a)^
({{8{data_in[51][4]}} & 8'h3f)^
({{8{data_in[51][5]}} & 8'h7e)^
({{8{data_in[51][6]}} & 8'hfc)^
({{8{data_in[51][7]}} & 8'hd3)^
({{8{data_in[52][0]}} & 8'h5)^
({{8{data_in[52][1]}} & 8'ha)^
({{8{data_in[52][2]}} & 8'h14)^
({{8{data_in[52][3]}} & 8'h28)^
({{8{data_in[52][4]}} & 8'h50)^
({{8{data_in[52][5]}} & 8'ha0)^
({{8{data_in[52][6]}} & 8'h6b)^
({{8{data_in[52][7]}} & 8'hd6)^
({{8{data_in[53][0]}} & 8'h3a)^
({{8{data_in[53][1]}} & 8'h74)^
({{8{data_in[53][2]}} & 8'he8)^
({{8{data_in[53][3]}} & 8'hfb)^
({{8{data_in[53][4]}} & 8'hdd)^
({{8{data_in[53][5]}} & 8'h91)^
({{8{data_in[53][6]}} & 8'h9)^
({{8{data_in[53][7]}} & 8'h12)^
({{8{data_in[54][0]}} & 8'ha7)^
({{8{data_in[54][1]}} & 8'h65)^
({{8{data_in[54][2]}} & 8'hca)^
({{8{data_in[54][3]}} & 8'hbf)^
({{8{data_in[54][4]}} & 8'h55)^
({{8{data_in[54][5]}} & 8'haa)^
({{8{data_in[54][6]}} & 8'h7f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[54][7]}} & 8'hfe)^
({{8{data_in[55][0]}} & 8'hed)^
({{8{data_in[55][1]}} & 8'hf1)^
({{8{data_in[55][2]}} & 8'hc9)^
({{8{data_in[55][3]}} & 8'hb9)^
({{8{data_in[55][4]}} & 8'h59)^
({{8{data_in[55][5]}} & 8'hb2)^
({{8{data_in[55][6]}} & 8'h4f)^
({{8{data_in[55][7]}} & 8'h9e)^
({{8{data_in[56][0]}} & 8'h65)^
({{8{data_in[56][1]}} & 8'hca)^
({{8{data_in[56][2]}} & 8'hbf)^
({{8{data_in[56][3]}} & 8'h55)^
({{8{data_in[56][4]}} & 8'haa)^
({{8{data_in[56][5]}} & 8'h7f)^
({{8{data_in[56][6]}} & 8'hfe)^
({{8{data_in[56][7]}} & 8'hd7)^
({{8{data_in[57][0]}} & 8'h3)^
({{8{data_in[57][1]}} & 8'h6)^
({{8{data_in[57][2]}} & 8'hc)^
({{8{data_in[57][3]}} & 8'h18)^
({{8{data_in[57][4]}} & 8'h30)^
({{8{data_in[57][5]}} & 8'h60)^
({{8{data_in[57][6]}} & 8'hc0)^
({{8{data_in[57][7]}} & 8'hab)^
({{8{data_in[58][0]}} & 8'hca)^
({{8{data_in[58][1]}} & 8'hbf)^
({{8{data_in[58][2]}} & 8'h55)^
({{8{data_in[58][3]}} & 8'haa)^
({{8{data_in[58][4]}} & 8'h7f)^
({{8{data_in[58][5]}} & 8'hfe)^
({{8{data_in[58][6]}} & 8'hd7)^
({{8{data_in[58][7]}} & 8'h85)^
({{8{data_in[59][0]}} & 8'h94)^
({{8{data_in[59][1]}} & 8'h3)^
({{8{data_in[59][2]}} & 8'h6)^
({{8{data_in[59][3]}} & 8'hc)^
({{8{data_in[59][4]}} & 8'h18)^
({{8{data_in[59][5]}} & 8'h30)^
({{8{data_in[59][6]}} & 8'h60)^
({{8{data_in[59][7]}} & 8'hc0)^
({{8{data_in[60][0]}} & 8'h9e)^
({{8{data_in[60][1]}} & 8'h17)^
({{8{data_in[60][2]}} & 8'h2e)^
({{8{data_in[60][3]}} & 8'h5c)^
({{8{data_in[60][4]}} & 8'hb8)^
({{8{data_in[60][5]}} & 8'h5b)^
({{8{data_in[60][6]}} & 8'hb6)^
({{8{data_in[60][7]}} & 8'h47)^
({{8{data_in[61][0]}} & 8'hc3)^
({{8{data_in[61][1]}} & 8'had)^
({{8{data_in[61][2]}} & 8'h71)^
({{8{data_in[61][3]}} & 8'he2)^
({{8{data_in[61][4]}} & 8'hef)^
({{8{data_in[61][5]}} & 8'hf5)^
({{8{data_in[61][6]}} & 8'hc1)^
({{8{data_in[61][7]}} & 8'ha9)^
({{8{data_in[62][0]}} & 8'ha3)^
({{8{data_in[62][1]}} & 8'h6d)^
({{8{data_in[62][2]}} & 8'hda)^
({{8{data_in[62][3]}} & 8'h9f)^
({{8{data_in[62][4]}} & 8'h15)^
({{8{data_in[62][5]}} & 8'h2a)^
({{8{data_in[62][6]}} & 8'h54)^
({{8{data_in[62][7]}} & 8'ha8)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[63][0]}} & 8'h23)^
({{8{data_in[63][1]}} & 8'h46)^
({{8{data_in[63][2]}} & 8'h8c)^
({{8{data_in[63][3]}} & 8'h33)^
({{8{data_in[63][4]}} & 8'h66)^
({{8{data_in[63][5]}} & 8'hcc)^
({{8{data_in[63][6]}} & 8'hb3)^
({{8{data_in[63][7]}} & 8'h4d)^
({{8{data_in[64][0]}} & 8'h54)^
({{8{data_in[64][1]}} & 8'ha8)^
({{8{data_in[64][2]}} & 8'h7b)^
({{8{data_in[64][3]}} & 8'hf6)^
({{8{data_in[64][4]}} & 8'hc7)^
({{8{data_in[64][5]}} & 8'ha5)^
({{8{data_in[64][6]}} & 8'h61)^
({{8{data_in[64][7]}} & 8'hc2)^
({{8{data_in[65][0]}} & 8'ha4)^
({{8{data_in[65][1]}} & 8'h63)^
({{8{data_in[65][2]}} & 8'hc6)^
({{8{data_in[65][3]}} & 8'ha7)^
({{8{data_in[65][4]}} & 8'h65)^
({{8{data_in[65][5]}} & 8'hca)^
({{8{data_in[65][6]}} & 8'hbf)^
({{8{data_in[65][7]}} & 8'h55)^
({{8{data_in[66][0]}} & 8'h4b)^
({{8{data_in[66][1]}} & 8'h96)^
({{8{data_in[66][2]}} & 8'h7)^
({{8{data_in[66][3]}} & 8'he)^
({{8{data_in[66][4]}} & 8'h1c)^
({{8{data_in[66][5]}} & 8'h38)^
({{8{data_in[66][6]}} & 8'h70)^
({{8{data_in[66][7]}} & 8'he0)^
({{8{data_in[67][0]}} & 8'h1c)^
({{8{data_in[67][1]}} & 8'h38)^
({{8{data_in[67][2]}} & 8'h70)^
({{8{data_in[67][3]}} & 8'he0)^
({{8{data_in[67][4]}} & 8'heb)^
({{8{data_in[67][5]}} & 8'hfd)^
({{8{data_in[67][6]}} & 8'hd1)^
({{8{data_in[67][7]}} & 8'h89)^
({{8{data_in[68][0]}} & 8'h84)^
({{8{data_in[68][1]}} & 8'h23)^
({{8{data_in[68][2]}} & 8'h46)^
({{8{data_in[68][3]}} & 8'h8c)^
({{8{data_in[68][4]}} & 8'h33)^
({{8{data_in[68][5]}} & 8'h66)^
({{8{data_in[68][6]}} & 8'hcc)^
({{8{data_in[68][7]}} & 8'hb3)^
({{8{data_in[69][0]}} & 8'h9a)^
({{8{data_in[69][1]}} & 8'h1f)^
({{8{data_in[69][2]}} & 8'h3e)^
({{8{data_in[69][3]}} & 8'h7c)^
({{8{data_in[69][4]}} & 8'hf8)^
({{8{data_in[69][5]}} & 8'hdb)^
({{8{data_in[69][6]}} & 8'h9d)^
({{8{data_in[69][7]}} & 8'h11)^
({{8{data_in[70][0]}} & 8'haa)^
({{8{data_in[70][1]}} & 8'h7f)^
({{8{data_in[70][2]}} & 8'hfe)^
({{8{data_in[70][3]}} & 8'hd7)^
({{8{data_in[70][4]}} & 8'h85)^
({{8{data_in[70][5]}} & 8'h21)^
({{8{data_in[70][6]}} & 8'h42)^
({{8{data_in[70][7]}} & 8'h84)^
({{8{data_in[71][0]}} & 8'he0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[71][1]}} & 8'heb)^
({{8{data_in[71][2]}} & 8'hfd)^
({{8{data_in[71][3]}} & 8'hd1)^
({{8{data_in[71][4]}} & 8'h89)^
({{8{data_in[71][5]}} & 8'h39)^
({{8{data_in[71][6]}} & 8'h72)^
({{8{data_in[71][7]}} & 8'he4)^
({{8{data_in[72][0]}} & 8'h5b)^
({{8{data_in[72][1]}} & 8'hb6)^
({{8{data_in[72][2]}} & 8'h47)^
({{8{data_in[72][3]}} & 8'h8e)^
({{8{data_in[72][4]}} & 8'h37)^
({{8{data_in[72][5]}} & 8'h6e)^
({{8{data_in[72][6]}} & 8'hdc)^
({{8{data_in[72][7]}} & 8'h93)^
({{8{data_in[73][0]}} & 8'hd2)^
({{8{data_in[73][1]}} & 8'h8f)^
({{8{data_in[73][2]}} & 8'h35)^
({{8{data_in[73][3]}} & 8'h6a)^
({{8{data_in[73][4]}} & 8'hd4)^
({{8{data_in[73][5]}} & 8'h83)^
({{8{data_in[73][6]}} & 8'h2d)^
({{8{data_in[73][7]}} & 8'h5a)^
({{8{data_in[74][0]}} & 8'h85)^
({{8{data_in[74][1]}} & 8'h21)^
({{8{data_in[74][2]}} & 8'h42)^
({{8{data_in[74][3]}} & 8'h84)^
({{8{data_in[74][4]}} & 8'h23)^
({{8{data_in[74][5]}} & 8'h46)^
({{8{data_in[74][6]}} & 8'h8c)^
({{8{data_in[74][7]}} & 8'h33)^
({{8{data_in[75][0]}} & 8'h96)^
({{8{data_in[75][1]}} & 8'h7)^
({{8{data_in[75][2]}} & 8'he)^
({{8{data_in[75][3]}} & 8'h1c)^
({{8{data_in[75][4]}} & 8'h38)^
({{8{data_in[75][5]}} & 8'h70)^
({{8{data_in[75][6]}} & 8'he0)^
({{8{data_in[75][7]}} & 8'heb)^
({{8{data_in[76][0]}} & 8'h1e)^
({{8{data_in[76][1]}} & 8'h3c)^
({{8{data_in[76][2]}} & 8'h78)^
({{8{data_in[76][3]}} & 8'hf0)^
({{8{data_in[76][4]}} & 8'hcb)^
({{8{data_in[76][5]}} & 8'hbd)^
({{8{data_in[76][6]}} & 8'h51)^
({{8{data_in[76][7]}} & 8'ha2)^
({{8{data_in[77][0]}} & 8'hdb)^
({{8{data_in[77][1]}} & 8'h9d)^
({{8{data_in[77][2]}} & 8'h11)^
({{8{data_in[77][3]}} & 8'h22)^
({{8{data_in[77][4]}} & 8'h44)^
({{8{data_in[77][5]}} & 8'h88)^
({{8{data_in[77][6]}} & 8'h3b)^
({{8{data_in[77][7]}} & 8'h76)^
({{8{data_in[78][0]}} & 8'he0)^
({{8{data_in[78][1]}} & 8'heb)^
({{8{data_in[78][2]}} & 8'hfd)^
({{8{data_in[78][3]}} & 8'hd1)^
({{8{data_in[78][4]}} & 8'h89)^
({{8{data_in[78][5]}} & 8'h39)^
({{8{data_in[78][6]}} & 8'h72)^
({{8{data_in[78][7]}} & 8'he4)^
({{8{data_in[79][0]}} & 8'h58)^
({{8{data_in[79][1]}} & 8'hb0})

```

Base 6.4 vs Base 6.3

```

({{8{data_in[79][2]}} & 8'h4b)^
({{8{data_in[79][3]}} & 8'h96)^
({{8{data_in[79][4]}} & 8'h7)^
({{8{data_in[79][5]}} & 8'he)^
({{8{data_in[79][6]}} & 8'h1c)^
({{8{data_in[79][7]}} & 8'h38)^
({{8{data_in[80][0]}} & 8'h64)^
({{8{data_in[80][1]}} & 8'hc8)^
({{8{data_in[80][2]}} & 8'hbb)^
({{8{data_in[80][3]}} & 8'h5d)^
({{8{data_in[80][4]}} & 8'hba)^
({{8{data_in[80][5]}} & 8'h5f)^
({{8{data_in[80][6]}} & 8'hbe)^
({{8{data_in[80][7]}} & 8'h57)^
({{8{data_in[81][0]}} & 8'h7)^
({{8{data_in[81][1]}} & 8'he)^
({{8{data_in[81][2]}} & 8'h1c)^
({{8{data_in[81][3]}} & 8'h38)^
({{8{data_in[81][4]}} & 8'h70)^
({{8{data_in[81][5]}} & 8'he0)^
({{8{data_in[81][6]}} & 8'heb)^
({{8{data_in[81][7]}} & 8'hfd)^
({{8{data_in[82][0]}} & 8'h63)^
({{8{data_in[82][1]}} & 8'hc6)^
({{8{data_in[82][2]}} & 8'ha7)^
({{8{data_in[82][3]}} & 8'h65)^
({{8{data_in[82][4]}} & 8'hca)^
({{8{data_in[82][5]}} & 8'hbf)^
({{8{data_in[82][6]}} & 8'h55)^
({{8{data_in[82][7]}} & 8'haa)^
({{8{data_in[83][0]}} & 8'h6b)^
({{8{data_in[83][1]}} & 8'hd6)^
({{8{data_in[83][2]}} & 8'h87)^
({{8{data_in[83][3]}} & 8'h25)^
({{8{data_in[83][4]}} & 8'h4a)^
({{8{data_in[83][5]}} & 8'h94)^
({{8{data_in[83][6]}} & 8'h3)^
({{8{data_in[83][7]}} & 8'h6)^
({{8{data_in[84][0]}} & 8'h7d)^
({{8{data_in[84][1]}} & 8'hfa)^
({{8{data_in[84][2]}} & 8'hdf)^
({{8{data_in[84][3]}} & 8'h95)^
({{8{data_in[84][4]}} & 8'h1)^
({{8{data_in[84][5]}} & 8'h2)^
({{8{data_in[84][6]}} & 8'h4)^
({{8{data_in[84][7]}} & 8'h8)^
({{8{data_in[85][0]}} & 8'h26)^
({{8{data_in[85][1]}} & 8'h4c)^
({{8{data_in[85][2]}} & 8'h98)^
({{8{data_in[85][3]}} & 8'h1b)^
({{8{data_in[85][4]}} & 8'h36)^
({{8{data_in[85][5]}} & 8'h6c)^
({{8{data_in[85][6]}} & 8'hd8)^
({{8{data_in[85][7]}} & 8'h9b)^
({{8{data_in[86][0]}} & 8'h6f)^
({{8{data_in[86][1]}} & 8'hde)^
({{8{data_in[86][2]}} & 8'h97)^
({{8{data_in[86][3]}} & 8'h5)^
({{8{data_in[86][4]}} & 8'ha)^
({{8{data_in[86][5]}} & 8'h14)^
({{8{data_in[86][6]}} & 8'h28)^
({{8{data_in[86][7]}} & 8'h50)^
({{8{data_in[87][0]}} & 8'h45)^
({{8{data_in[87][1]}} & 8'h8a)^
({{8{data_in[87][2]}} & 8'h3f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[87][3]}} & 8'h7e)^
({{8{data_in[87][4]}} & 8'hfc)^
({{8{data_in[87][5]}} & 8'hd3)^
({{8{data_in[87][6]}} & 8'h8d)^
({{8{data_in[87][7]}} & 8'h31)^
({{8{data_in[88][0]}} & 8'he)^
({{8{data_in[88][1]}} & 8'h1c)^
({{8{data_in[88][2]}} & 8'h38)^
({{8{data_in[88][3]}} & 8'h70)^
({{8{data_in[88][4]}} & 8'he0)^
({{8{data_in[88][5]}} & 8'heb)^
({{8{data_in[88][6]}} & 8'hfd)^
({{8{data_in[88][7]}} & 8'hd1)^
({{8{data_in[89][0]}} & 8'h63)^
({{8{data_in[89][1]}} & 8'hc6)^
({{8{data_in[89][2]}} & 8'ha7)^
({{8{data_in[89][3]}} & 8'h65)^
({{8{data_in[89][4]}} & 8'hca)^
({{8{data_in[89][5]}} & 8'hbf)^
({{8{data_in[89][6]}} & 8'h55)^
({{8{data_in[89][7]}} & 8'haa)^
({{8{data_in[90][0]}} & 8'h5f)^
({{8{data_in[90][1]}} & 8'hbe)^
({{8{data_in[90][2]}} & 8'h57)^
({{8{data_in[90][3]}} & 8'hae)^
({{8{data_in[90][4]}} & 8'h77)^
({{8{data_in[90][5]}} & 8'hee)^
({{8{data_in[90][6]}} & 8'hf7)^
({{8{data_in[90][7]}} & 8'hc5)^
({{8{data_in[91][0]}} & 8'h33)^
({{8{data_in[91][1]}} & 8'h66)^
({{8{data_in[91][2]}} & 8'hcc)^
({{8{data_in[91][3]}} & 8'hb3)^
({{8{data_in[91][4]}} & 8'h4d)^
({{8{data_in[91][5]}} & 8'h9a)^
({{8{data_in[91][6]}} & 8'h1f)^
({{8{data_in[91][7]}} & 8'h3e)^
({{8{data_in[92][0]}} & 8'h22)^
({{8{data_in[92][1]}} & 8'h44)^
({{8{data_in[92][2]}} & 8'h88)^
({{8{data_in[92][3]}} & 8'h3b)^
({{8{data_in[92][4]}} & 8'h76)^
({{8{data_in[92][5]}} & 8'hec)^
({{8{data_in[92][6]}} & 8'hf3)^
({{8{data_in[92][7]}} & 8'hcd)^
({{8{data_in[93][0]}} & 8'h43)^
({{8{data_in[93][1]}} & 8'h86)^
({{8{data_in[93][2]}} & 8'h27)^
({{8{data_in[93][3]}} & 8'h4e)^
({{8{data_in[93][4]}} & 8'h9c)^
({{8{data_in[93][5]}} & 8'h13)^
({{8{data_in[93][6]}} & 8'h26)^
({{8{data_in[93][7]}} & 8'h4c)^
({{8{data_in[94][0]}} & 8'hcf)^
({{8{data_in[94][1]}} & 8'hb5)^
({{8{data_in[94][2]}} & 8'h41)^
({{8{data_in[94][3]}} & 8'h82)^
({{8{data_in[94][4]}} & 8'h2f)^
({{8{data_in[94][5]}} & 8'h5e)^
({{8{data_in[94][6]}} & 8'hbc)^
({{8{data_in[94][7]}} & 8'h53)^
({{8{data_in[95][0]}} & 8'h54)^
({{8{data_in[95][1]}} & 8'ha8)^
({{8{data_in[95][2]}} & 8'h7b)^
({{8{data_in[95][3]}} & 8'hf6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[95][4]}} & 8'hc7)^
({{8{data_in[95][5]}} & 8'ha5)^
({{8{data_in[95][6]}} & 8'h61)^
({{8{data_in[95][7]}} & 8'hc2)^
({{8{data_in[96][0]}} & 8'hd0)^
({{8{data_in[96][1]}} & 8'h8b)^
({{8{data_in[96][2]}} & 8'h3d)^
({{8{data_in[96][3]}} & 8'h7a)^
({{8{data_in[96][4]}} & 8'hf4)^
({{8{data_in[96][5]}} & 8'hc3)^
({{8{data_in[96][6]}} & 8'had)^
({{8{data_in[96][7]}} & 8'h71)^
({{8{data_in[97][0]}} & 8'h57)^
({{8{data_in[97][1]}} & 8'hae)^
({{8{data_in[97][2]}} & 8'h77)^
({{8{data_in[97][3]}} & 8'hee)^
({{8{data_in[97][4]}} & 8'hf7)^
({{8{data_in[97][5]}} & 8'hc5)^
({{8{data_in[97][6]}} & 8'ha1)^
({{8{data_in[97][7]}} & 8'h69)^
({{8{data_in[98][0]}} & 8'hcb)^
({{8{data_in[98][1]}} & 8'hbd)^
({{8{data_in[98][2]}} & 8'h51)^
({{8{data_in[98][3]}} & 8'ha2)^
({{8{data_in[98][4]}} & 8'h6f)^
({{8{data_in[98][5]}} & 8'hde)^
({{8{data_in[98][6]}} & 8'h97)^
({{8{data_in[98][7]}} & 8'h5)^
({{8{data_in[99][0]}} & 8'h7f)^
({{8{data_in[99][1]}} & 8'hfe)^
({{8{data_in[99][2]}} & 8'hd7)^
({{8{data_in[99][3]}} & 8'h85)^
({{8{data_in[99][4]}} & 8'h21)^
({{8{data_in[99][5]}} & 8'h42)^
({{8{data_in[99][6]}} & 8'h84)^
({{8{data_in[99][7]}} & 8'h23)^
({{8{data_in[100][0]}} & 8'hb9)^
({{8{data_in[100][1]}} & 8'h59)^
({{8{data_in[100][2]}} & 8'hb2)^
({{8{data_in[100][3]}} & 8'h4f)^
({{8{data_in[100][4]}} & 8'h9e)^
({{8{data_in[100][5]}} & 8'h17)^
({{8{data_in[100][6]}} & 8'h2e)^
({{8{data_in[100][7]}} & 8'h5c)^
({{8{data_in[101][0]}} & 8'h42)^
({{8{data_in[101][1]}} & 8'h84)^
({{8{data_in[101][2]}} & 8'h23)^
({{8{data_in[101][3]}} & 8'h46)^
({{8{data_in[101][4]}} & 8'h8c)^
({{8{data_in[101][5]}} & 8'h33)^
({{8{data_in[101][6]}} & 8'h66)^
({{8{data_in[101][7]}} & 8'hcc)^
({{8{data_in[102][0]}} & 8'h5a)^
({{8{data_in[102][1]}} & 8'hb4)^
({{8{data_in[102][2]}} & 8'h43)^
({{8{data_in[102][3]}} & 8'h86)^
({{8{data_in[102][4]}} & 8'h27)^
({{8{data_in[102][5]}} & 8'h4e)^
({{8{data_in[102][6]}} & 8'h9c)^
({{8{data_in[102][7]}} & 8'h13)^
({{8{data_in[103][0]}} & 8'hd2)^
({{8{data_in[103][1]}} & 8'h8f)^
({{8{data_in[103][2]}} & 8'h35)^
({{8{data_in[103][3]}} & 8'h6a)^
({{8{data_in[103][4]}} & 8'hd4)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[103][5]}} & 8'h83)^
({{8{data_in[103][6]}} & 8'h2d)^
({{8{data_in[103][7]}} & 8'h5a)^
({{8{data_in[104][0]}} & 8'h56)^
({{8{data_in[104][1]}} & 8'hac)^
({{8{data_in[104][2]}} & 8'h73)^
({{8{data_in[104][3]}} & 8'he6)^
({{8{data_in[104][4]}} & 8'he7)^
({{8{data_in[104][5]}} & 8'he5)^
({{8{data_in[104][6]}} & 8'he1)^
({{8{data_in[104][7]}} & 8'he9)^
({{8{data_in[105][0]}} & 8'h34)^
({{8{data_in[105][1]}} & 8'h68)^
({{8{data_in[105][2]}} & 8'hd0)^
({{8{data_in[105][3]}} & 8'h8b)^
({{8{data_in[105][4]}} & 8'h3d)^
({{8{data_in[105][5]}} & 8'h7a)^
({{8{data_in[105][6]}} & 8'hf4)^
({{8{data_in[105][7]}} & 8'hc3)^
({{8{data_in[106][0]}} & 8'hd7)^
({{8{data_in[106][1]}} & 8'h85)^
({{8{data_in[106][2]}} & 8'h21)^
({{8{data_in[106][3]}} & 8'h42)^
({{8{data_in[106][4]}} & 8'h84)^
({{8{data_in[106][5]}} & 8'h23)^
({{8{data_in[106][6]}} & 8'h46)^
({{8{data_in[106][7]}} & 8'h8c)^
({{8{data_in[107][0]}} & 8'hbf)^
({{8{data_in[107][1]}} & 8'h55)^
({{8{data_in[107][2]}} & 8'haa)^
({{8{data_in[107][3]}} & 8'h7f)^
({{8{data_in[107][4]}} & 8'hfe)^
({{8{data_in[107][5]}} & 8'hd7)^
({{8{data_in[107][6]}} & 8'h85)^
({{8{data_in[107][7]}} & 8'h21)^
({{8{data_in[108][0]}} & 8'he4)^
({{8{data_in[108][1]}} & 8'he3)^
({{8{data_in[108][2]}} & 8'hed)^
({{8{data_in[108][3]}} & 8'hf1)^
({{8{data_in[108][4]}} & 8'hc9)^
({{8{data_in[108][5]}} & 8'hb9)^
({{8{data_in[108][6]}} & 8'h59)^
({{8{data_in[108][7]}} & 8'hb2)^
({{8{data_in[109][0]}} & 8'h6d)^
({{8{data_in[109][1]}} & 8'hda)^
({{8{data_in[109][2]}} & 8'h9f)^
({{8{data_in[109][3]}} & 8'h15)^
({{8{data_in[109][4]}} & 8'h2a)^
({{8{data_in[109][5]}} & 8'h54)^
({{8{data_in[109][6]}} & 8'ha8)^
({{8{data_in[109][7]}} & 8'h7b)^
({{8{data_in[110][0]}} & 8'h19)^
({{8{data_in[110][1]}} & 8'h32)^
({{8{data_in[110][2]}} & 8'h64)^
({{8{data_in[110][3]}} & 8'hc8)^
({{8{data_in[110][4]}} & 8'hbb)^
({{8{data_in[110][5]}} & 8'h5d)^
({{8{data_in[110][6]}} & 8'hba)^
({{8{data_in[110][7]}} & 8'h5f)^
({{8{data_in[111][0]}} & 8'h4a)^
({{8{data_in[111][1]}} & 8'h94)^
({{8{data_in[111][2]}} & 8'h3)^
({{8{data_in[111][3]}} & 8'h6)^
({{8{data_in[111][4]}} & 8'hc)^
({{8{data_in[111][5]}} & 8'h18)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[111][6]}} & 8'h30)^
({{8{data_in[111][7]}} & 8'h60)^
({{8{data_in[112][0]}} & 8'hde)^
({{8{data_in[112][1]}} & 8'h97)^
({{8{data_in[112][2]}} & 8'h5)^
({{8{data_in[112][3]}} & 8'ha)^
({{8{data_in[112][4]}} & 8'h14)^
({{8{data_in[112][5]}} & 8'h28)^
({{8{data_in[112][6]}} & 8'h50)^
({{8{data_in[112][7]}} & 8'ha0)^
({{8{data_in[113][0]}} & 8'haf)^
({{8{data_in[113][1]}} & 8'h75)^
({{8{data_in[113][2]}} & 8'hea)^
({{8{data_in[113][3]}} & 8'hff)^
({{8{data_in[113][4]}} & 8'hd5)^
({{8{data_in[113][5]}} & 8'h81)^
({{8{data_in[113][6]}} & 8'h29)^
({{8{data_in[113][7]}} & 8'h52)^
({{8{data_in[114][0]}} & 8'h51)^
({{8{data_in[114][1]}} & 8'ha2)^
({{8{data_in[114][2]}} & 8'h6f)^
({{8{data_in[114][3]}} & 8'hde)^
({{8{data_in[114][4]}} & 8'h97)^
({{8{data_in[114][5]}} & 8'h5)^
({{8{data_in[114][6]}} & 8'ha)^
({{8{data_in[114][7]}} & 8'h14)^
({{8{data_in[115][0]}} & 8'h63)^
({{8{data_in[115][1]}} & 8'hc6)^
({{8{data_in[115][2]}} & 8'ha7)^
({{8{data_in[115][3]}} & 8'h65)^
({{8{data_in[115][4]}} & 8'hca)^
({{8{data_in[115][5]}} & 8'hbf)^
({{8{data_in[115][6]}} & 8'h55)^
({{8{data_in[115][7]}} & 8'haa)^
({{8{data_in[116][0]}} & 8'h69)^
({{8{data_in[116][1]}} & 8'hd2)^
({{8{data_in[116][2]}} & 8'h8f)^
({{8{data_in[116][3]}} & 8'h35)^
({{8{data_in[116][4]}} & 8'h6a)^
({{8{data_in[116][5]}} & 8'hd4)^
({{8{data_in[116][6]}} & 8'h83)^
({{8{data_in[116][7]}} & 8'h2d)^
({{8{data_in[117][0]}} & 8'h64)^
({{8{data_in[117][1]}} & 8'hc8)^
({{8{data_in[117][2]}} & 8'hbb)^
({{8{data_in[117][3]}} & 8'h5d)^
({{8{data_in[117][4]}} & 8'hba)^
({{8{data_in[117][5]}} & 8'h5f)^
({{8{data_in[117][6]}} & 8'hbe)^
({{8{data_in[117][7]}} & 8'h57)^
({{8{data_in[118][0]}} & 8'h51)^
({{8{data_in[118][1]}} & 8'ha2)^
({{8{data_in[118][2]}} & 8'h6f)^
({{8{data_in[118][3]}} & 8'hde)^
({{8{data_in[118][4]}} & 8'h97)^
({{8{data_in[118][5]}} & 8'h5)^
({{8{data_in[118][6]}} & 8'ha)^
({{8{data_in[118][7]}} & 8'h14)^
({{8{data_in[119][0]}} & 8'hf1)^
({{8{data_in[119][1]}} & 8'hc9)^
({{8{data_in[119][2]}} & 8'hb9)^
({{8{data_in[119][3]}} & 8'h59)^
({{8{data_in[119][4]}} & 8'hb2)^
({{8{data_in[119][5]}} & 8'h4f)^
({{8{data_in[119][6]}} & 8'h9e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[119][7]}} & 8'h17)^
({{8{data_in[120][0]}} & 8'h2f)^
({{8{data_in[120][1]}} & 8'h5e)^
({{8{data_in[120][2]}} & 8'hbc)^
({{8{data_in[120][3]}} & 8'h53)^
({{8{data_in[120][4]}} & 8'ha6)^
({{8{data_in[120][5]}} & 8'h67)^
({{8{data_in[120][6]}} & 8'hce)^
({{8{data_in[120][7]}} & 8'hb7)^
({{8{data_in[121][0]}} & 8'h67)^
({{8{data_in[121][1]}} & 8'hce)^
({{8{data_in[121][2]}} & 8'hb7)^
({{8{data_in[121][3]}} & 8'h45)^
({{8{data_in[121][4]}} & 8'h8a)^
({{8{data_in[121][5]}} & 8'h3f)^
({{8{data_in[121][6]}} & 8'h7e)^
({{8{data_in[121][7]}} & 8'hfc)^
({{8{data_in[122][0]}} & 8'hd9)^
({{8{data_in[122][1]}} & 8'h99)^
({{8{data_in[122][2]}} & 8'h19)^
({{8{data_in[122][3]}} & 8'h32)^
({{8{data_in[122][4]}} & 8'h64)^
({{8{data_in[122][5]}} & 8'hc8)^
({{8{data_in[122][6]}} & 8'hbb)^
({{8{data_in[122][7]}} & 8'h5d)^
({{8{data_in[123][0]}} & 8'hd8)^
({{8{data_in[123][1]}} & 8'h9b)^
({{8{data_in[123][2]}} & 8'h1d)^
({{8{data_in[123][3]}} & 8'h3a)^
({{8{data_in[123][4]}} & 8'h74)^
({{8{data_in[123][5]}} & 8'he8)^
({{8{data_in[123][6]}} & 8'hfb)^
({{8{data_in[123][7]}} & 8'hdd)^
({{8{data_in[124][0]}} & 8'h18)^
({{8{data_in[124][1]}} & 8'h30)^
({{8{data_in[124][2]}} & 8'h60)^
({{8{data_in[124][3]}} & 8'hc0)^
({{8{data_in[124][4]}} & 8'hab)^
({{8{data_in[124][5]}} & 8'h7d)^
({{8{data_in[124][6]}} & 8'hfa)^
({{8{data_in[124][7]}} & 8'hdf)^
({{8{data_in[125][0]}} & 8'h2d)^
({{8{data_in[125][1]}} & 8'h5a)^
({{8{data_in[125][2]}} & 8'hb4)^
({{8{data_in[125][3]}} & 8'h43)^
({{8{data_in[125][4]}} & 8'h86)^
({{8{data_in[125][5]}} & 8'h27)^
({{8{data_in[125][6]}} & 8'h4e)^
({{8{data_in[125][7]}} & 8'h9c)^
({{8{data_in[126][0]}} & 8'h43)^
({{8{data_in[126][1]}} & 8'h86)^
({{8{data_in[126][2]}} & 8'h27)^
({{8{data_in[126][3]}} & 8'h4e)^
({{8{data_in[126][4]}} & 8'h9c)^
({{8{data_in[126][5]}} & 8'h13)^
({{8{data_in[126][6]}} & 8'h26)^
({{8{data_in[126][7]}} & 8'h4c)^
({{8{data_in[127][0]}} & 8'h90)^
({{8{data_in[127][1]}} & 8'hb)^
({{8{data_in[127][2]}} & 8'h16)^
({{8{data_in[127][3]}} & 8'h2c)^
({{8{data_in[127][4]}} & 8'h58)^
({{8{data_in[127][5]}} & 8'hb0)^
({{8{data_in[127][6]}} & 8'h4b)^
({{8{data_in[127][7]}} & 8'h96)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[128][0]}} & 8'hcc)^
({{8{data_in[128][1]}} & 8'hb3)^
({{8{data_in[128][2]}} & 8'h4d)^
({{8{data_in[128][3]}} & 8'h9a)^
({{8{data_in[128][4]}} & 8'h1f)^
({{8{data_in[128][5]}} & 8'h3e)^
({{8{data_in[128][6]}} & 8'h7c)^
({{8{data_in[128][7]}} & 8'hf8)^
({{8{data_in[129][0]}} & 8'h71)^
({{8{data_in[129][1]}} & 8'he2)^
({{8{data_in[129][2]}} & 8'hef)^
({{8{data_in[129][3]}} & 8'hf5)^
({{8{data_in[129][4]}} & 8'hc1)^
({{8{data_in[129][5]}} & 8'ha9)^
({{8{data_in[129][6]}} & 8'h79)^
({{8{data_in[129][7]}} & 8'hf2)^
({{8{data_in[130][0]}} & 8'h9d)^
({{8{data_in[130][1]}} & 8'h11)^
({{8{data_in[130][2]}} & 8'h22)^
({{8{data_in[130][3]}} & 8'h44)^
({{8{data_in[130][4]}} & 8'h88)^
({{8{data_in[130][5]}} & 8'h3b)^
({{8{data_in[130][6]}} & 8'h76)^
({{8{data_in[130][7]}} & 8'hec)^
({{8{data_in[131][0]}} & 8'h3e)^
({{8{data_in[131][1]}} & 8'h7c)^
({{8{data_in[131][2]}} & 8'hf8)^
({{8{data_in[131][3]}} & 8'hdb)^
({{8{data_in[131][4]}} & 8'h9d)^
({{8{data_in[131][5]}} & 8'h11)^
({{8{data_in[131][6]}} & 8'h22)^
({{8{data_in[131][7]}} & 8'h44)^
({{8{data_in[132][0]}} & 8'hf4)^
({{8{data_in[132][1]}} & 8'hc3)^
({{8{data_in[132][2]}} & 8'had)^
({{8{data_in[132][3]}} & 8'h71)^
({{8{data_in[132][4]}} & 8'he2)^
({{8{data_in[132][5]}} & 8'hef)^
({{8{data_in[132][6]}} & 8'hf5)^
({{8{data_in[132][7]}} & 8'hc1)^
({{8{data_in[133][0]}} & 8'h40)^
({{8{data_in[133][1]}} & 8'h80)^
({{8{data_in[133][2]}} & 8'h2b)^
({{8{data_in[133][3]}} & 8'h56)^
({{8{data_in[133][4]}} & 8'hac)^
({{8{data_in[133][5]}} & 8'h73)^
({{8{data_in[133][6]}} & 8'he6)^
({{8{data_in[133][7]}} & 8'he7)^
({{8{data_in[134][0]}} & 8'h5d)^
({{8{data_in[134][1]}} & 8'hba)^
({{8{data_in[134][2]}} & 8'h5f)^
({{8{data_in[134][3]}} & 8'hbe)^
({{8{data_in[134][4]}} & 8'h57)^
({{8{data_in[134][5]}} & 8'hae)^
({{8{data_in[134][6]}} & 8'h77)^
({{8{data_in[134][7]}} & 8'hee)^
({{8{data_in[135][0]}} & 8'hc8)^
({{8{data_in[135][1]}} & 8'hbb)^
({{8{data_in[135][2]}} & 8'h5d)^
({{8{data_in[135][3]}} & 8'hba)^
({{8{data_in[135][4]}} & 8'h5f)^
({{8{data_in[135][5]}} & 8'hbe)^
({{8{data_in[135][6]}} & 8'h57)^
({{8{data_in[135][7]}} & 8'hae)^
({{8{data_in[136][0]}} & 8'h54)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[136][1]}} & 8'ha8)^
({{8{data_in[136][2]}} & 8'h7b)^
({{8{data_in[136][3]}} & 8'hf6)^
({{8{data_in[136][4]}} & 8'hc7)^
({{8{data_in[136][5]}} & 8'ha5)^
({{8{data_in[136][6]}} & 8'h61)^
({{8{data_in[136][7]}} & 8'hc2)^
({{8{data_in[137][0]}} & 8'h6c)^
({{8{data_in[137][1]}} & 8'hd8)^
({{8{data_in[137][2]}} & 8'h9b)^
({{8{data_in[137][3]}} & 8'h1d)^
({{8{data_in[137][4]}} & 8'h3a)^
({{8{data_in[137][5]}} & 8'h74)^
({{8{data_in[137][6]}} & 8'he8)^
({{8{data_in[137][7]}} & 8'hfb)^
({{8{data_in[138][0]}} & 8'h5e)^
({{8{data_in[138][1]}} & 8'hbc)^
({{8{data_in[138][2]}} & 8'h53)^
({{8{data_in[138][3]}} & 8'ha6)^
({{8{data_in[138][4]}} & 8'h67)^
({{8{data_in[138][5]}} & 8'hce)^
({{8{data_in[138][6]}} & 8'hb7)^
({{8{data_in[138][7]}} & 8'h45)^
({{8{data_in[139][0]}} & 8'hc2)^
({{8{data_in[139][1]}} & 8'haf)^
({{8{data_in[139][2]}} & 8'h75)^
({{8{data_in[139][3]}} & 8'hea)^
({{8{data_in[139][4]}} & 8'hff)^
({{8{data_in[139][5]}} & 8'hd5)^
({{8{data_in[139][6]}} & 8'h81)^
({{8{data_in[139][7]}} & 8'h29)^
({{8{data_in[140][0]}} & 8'h92)^
({{8{data_in[140][1]}} & 8'hf)^
({{8{data_in[140][2]}} & 8'h1e)^
({{8{data_in[140][3]}} & 8'h3c)^
({{8{data_in[140][4]}} & 8'h78)^
({{8{data_in[140][5]}} & 8'hf0)^
({{8{data_in[140][6]}} & 8'hcb)^
({{8{data_in[140][7]}} & 8'hbd)^
({{8{data_in[141][0]}} & 8'h83)^
({{8{data_in[141][1]}} & 8'h2d)^
({{8{data_in[141][2]}} & 8'h5a)^
({{8{data_in[141][3]}} & 8'hb4)^
({{8{data_in[141][4]}} & 8'h43)^
({{8{data_in[141][5]}} & 8'h86)^
({{8{data_in[141][6]}} & 8'h27)^
({{8{data_in[141][7]}} & 8'h4e)^
({{8{data_in[142][0]}} & 8'hb0)^
({{8{data_in[142][1]}} & 8'h4b)^
({{8{data_in[142][2]}} & 8'h96)^
({{8{data_in[142][3]}} & 8'h7)^
({{8{data_in[142][4]}} & 8'he)^
({{8{data_in[142][5]}} & 8'h1c)^
({{8{data_in[142][6]}} & 8'h38)^
({{8{data_in[142][7]}} & 8'h70)^
({{8{data_in[143][0]}} & 8'h35)^
({{8{data_in[143][1]}} & 8'h6a)^
({{8{data_in[143][2]}} & 8'hd4)^
({{8{data_in[143][3]}} & 8'h83)^
({{8{data_in[143][4]}} & 8'h2d)^
({{8{data_in[143][5]}} & 8'h5a)^
({{8{data_in[143][6]}} & 8'hb4)^
({{8{data_in[143][7]}} & 8'h43)^
({{8{data_in[144][0]}} & 8'h4d)^
({{8{data_in[144][1]}} & 8'h9a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[144][2]}} & 8'h1f)^
({{8{data_in[144][3]}} & 8'h3e)^
({{8{data_in[144][4]}} & 8'h7c)^
({{8{data_in[144][5]}} & 8'hf8)^
({{8{data_in[144][6]}} & 8'hdb)^
({{8{data_in[144][7]}} & 8'h9d)^
({{8{data_in[145][0]}} & 8'h59)^
({{8{data_in[145][1]}} & 8'hb2)^
({{8{data_in[145][2]}} & 8'h4f)^
({{8{data_in[145][3]}} & 8'h9e)^
({{8{data_in[145][4]}} & 8'h17)^
({{8{data_in[145][5]}} & 8'h2e)^
({{8{data_in[145][6]}} & 8'h5c)^
({{8{data_in[145][7]}} & 8'hb8)^
({{8{data_in[146][0]}} & 8'h58)^
({{8{data_in[146][1]}} & 8'hb0)^
({{8{data_in[146][2]}} & 8'h4b)^
({{8{data_in[146][3]}} & 8'h96)^
({{8{data_in[146][4]}} & 8'h7)^
({{8{data_in[146][5]}} & 8'he)^
({{8{data_in[146][6]}} & 8'h1c)^
({{8{data_in[146][7]}} & 8'h38)^
({{8{data_in[147][0]}} & 8'hc4)^
({{8{data_in[147][1]}} & 8'ha3)^
({{8{data_in[147][2]}} & 8'h6d)^
({{8{data_in[147][3]}} & 8'hda)^
({{8{data_in[147][4]}} & 8'h9f)^
({{8{data_in[147][5]}} & 8'h15)^
({{8{data_in[147][6]}} & 8'h2a)^
({{8{data_in[147][7]}} & 8'h54)^
({{8{data_in[148][0]}} & 8'h35)^
({{8{data_in[148][1]}} & 8'h6a)^
({{8{data_in[148][2]}} & 8'hd4)^
({{8{data_in[148][3]}} & 8'h83)^
({{8{data_in[148][4]}} & 8'h2d)^
({{8{data_in[148][5]}} & 8'h5a)^
({{8{data_in[148][6]}} & 8'hb4)^
({{8{data_in[148][7]}} & 8'h43)^
({{8{data_in[149][0]}} & 8'h85)^
({{8{data_in[149][1]}} & 8'h21)^
({{8{data_in[149][2]}} & 8'h42)^
({{8{data_in[149][3]}} & 8'h84)^
({{8{data_in[149][4]}} & 8'h23)^
({{8{data_in[149][5]}} & 8'h46)^
({{8{data_in[149][6]}} & 8'h8c)^
({{8{data_in[149][7]}} & 8'h33)^
({{8{data_in[150][0]}} & 8'h3)^
({{8{data_in[150][1]}} & 8'h6)^
({{8{data_in[150][2]}} & 8'hc)^
({{8{data_in[150][3]}} & 8'h18)^
({{8{data_in[150][4]}} & 8'h30)^
({{8{data_in[150][5]}} & 8'h60)^
({{8{data_in[150][6]}} & 8'hc0)^
({{8{data_in[150][7]}} & 8'hab)^
({{8{data_in[151][0]}} & 8'he8)^
({{8{data_in[151][1]}} & 8'hfb)^
({{8{data_in[151][2]}} & 8'hdd)^
({{8{data_in[151][3]}} & 8'h91)^
({{8{data_in[151][4]}} & 8'h9)^
({{8{data_in[151][5]}} & 8'h12)^
({{8{data_in[151][6]}} & 8'h24)^
({{8{data_in[151][7]}} & 8'h48)^
({{8{data_in[152][0]}} & 8'hb5)^
({{8{data_in[152][1]}} & 8'h41)^
({{8{data_in[152][2]}} & 8'h82)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[152][3]}} & 8'h2f)^
({{8{data_in[152][4]}} & 8'h5e)^
({{8{data_in[152][5]}} & 8'hbc)^
({{8{data_in[152][6]}} & 8'h53)^
({{8{data_in[152][7]}} & 8'ha6)^
({{8{data_in[153][0]}} & 8'h12)^
({{8{data_in[153][1]}} & 8'h24)^
({{8{data_in[153][2]}} & 8'h48)^
({{8{data_in[153][3]}} & 8'h90)^
({{8{data_in[153][4]}} & 8'hb)^
({{8{data_in[153][5]}} & 8'h16)^
({{8{data_in[153][6]}} & 8'h2c)^
({{8{data_in[153][7]}} & 8'h58)^
({{8{data_in[154][0]}} & 8'hb4)^
({{8{data_in[154][1]}} & 8'h43)^
({{8{data_in[154][2]}} & 8'h86)^
({{8{data_in[154][3]}} & 8'h27)^
({{8{data_in[154][4]}} & 8'h4e)^
({{8{data_in[154][5]}} & 8'h9c)^
({{8{data_in[154][6]}} & 8'h13)^
({{8{data_in[154][7]}} & 8'h26)^
({{8{data_in[155][0]}} & 8'h5)^
({{8{data_in[155][1]}} & 8'ha)^
({{8{data_in[155][2]}} & 8'h14)^
({{8{data_in[155][3]}} & 8'h28)^
({{8{data_in[155][4]}} & 8'h50)^
({{8{data_in[155][5]}} & 8'ha0)^
({{8{data_in[155][6]}} & 8'h6b)^
({{8{data_in[155][7]}} & 8'hd6)^
({{8{data_in[156][0]}} & 8'h71)^
({{8{data_in[156][1]}} & 8'he2)^
({{8{data_in[156][2]}} & 8'hef)^
({{8{data_in[156][3]}} & 8'hf5)^
({{8{data_in[156][4]}} & 8'hc1)^
({{8{data_in[156][5]}} & 8'ha9)^
({{8{data_in[156][6]}} & 8'h79)^
({{8{data_in[156][7]}} & 8'hf2)^
({{8{data_in[157][0]}} & 8'hd8)^
({{8{data_in[157][1]}} & 8'h9b)^
({{8{data_in[157][2]}} & 8'h1d)^
({{8{data_in[157][3]}} & 8'h3a)^
({{8{data_in[157][4]}} & 8'h74)^
({{8{data_in[157][5]}} & 8'he8)^
({{8{data_in[157][6]}} & 8'hfb)^
({{8{data_in[157][7]}} & 8'hdd)^
({{8{data_in[158][0]}} & 8'h2e)^
({{8{data_in[158][1]}} & 8'h5c)^
({{8{data_in[158][2]}} & 8'hb8)^
({{8{data_in[158][3]}} & 8'h5b)^
({{8{data_in[158][4]}} & 8'hb6)^
({{8{data_in[158][5]}} & 8'h47)^
({{8{data_in[158][6]}} & 8'h8e)^
({{8{data_in[158][7]}} & 8'h37)^
({{8{data_in[159][0]}} & 8'h2e)^
({{8{data_in[159][1]}} & 8'h5c)^
({{8{data_in[159][2]}} & 8'hb8)^
({{8{data_in[159][3]}} & 8'h5b)^
({{8{data_in[159][4]}} & 8'hb6)^
({{8{data_in[159][5]}} & 8'h47)^
({{8{data_in[159][6]}} & 8'h8e)^
({{8{data_in[159][7]}} & 8'h37)^
({{8{data_in[160][0]}} & 8'hf3)^
({{8{data_in[160][1]}} & 8'hcd)^
({{8{data_in[160][2]}} & 8'hb1)^
({{8{data_in[160][3]}} & 8'h49)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[160][4]}} & 8'h92)^
({{8{data_in[160][5]}} & 8'hf)^
({{8{data_in[160][6]}} & 8'h1e)^
({{8{data_in[160][7]}} & 8'h3c)^
({{8{data_in[161][0]}} & 8'h1e)^
({{8{data_in[161][1]}} & 8'h3c)^
({{8{data_in[161][2]}} & 8'h78)^
({{8{data_in[161][3]}} & 8'hf0)^
({{8{data_in[161][4]}} & 8'hcb)^
({{8{data_in[161][5]}} & 8'hbd)^
({{8{data_in[161][6]}} & 8'h51)^
({{8{data_in[161][7]}} & 8'ha2)^
({{8{data_in[162][0]}} & 8'h24)^
({{8{data_in[162][1]}} & 8'h48)^
({{8{data_in[162][2]}} & 8'h90)^
({{8{data_in[162][3]}} & 8'hb)^
({{8{data_in[162][4]}} & 8'h16)^
({{8{data_in[162][5]}} & 8'h2c)^
({{8{data_in[162][6]}} & 8'h58)^
({{8{data_in[162][7]}} & 8'hb0)^
({{8{data_in[163][0]}} & 8'hea)^
({{8{data_in[163][1]}} & 8'hff)^
({{8{data_in[163][2]}} & 8'hd5)^
({{8{data_in[163][3]}} & 8'h81)^
({{8{data_in[163][4]}} & 8'h29)^
({{8{data_in[163][5]}} & 8'h52)^
({{8{data_in[163][6]}} & 8'ha4)^
({{8{data_in[163][7]}} & 8'h63)^
({{8{data_in[164][0]}} & 8'hac)^
({{8{data_in[164][1]}} & 8'h73)^
({{8{data_in[164][2]}} & 8'he6)^
({{8{data_in[164][3]}} & 8'he7)^
({{8{data_in[164][4]}} & 8'he5)^
({{8{data_in[164][5]}} & 8'he1)^
({{8{data_in[164][6]}} & 8'he9)^
({{8{data_in[164][7]}} & 8'hf9)^
({{8{data_in[165][0]}} & 8'h24)^
({{8{data_in[165][1]}} & 8'h48)^
({{8{data_in[165][2]}} & 8'h90)^
({{8{data_in[165][3]}} & 8'hb)^
({{8{data_in[165][4]}} & 8'h16)^
({{8{data_in[165][5]}} & 8'h2c)^
({{8{data_in[165][6]}} & 8'h58)^
({{8{data_in[165][7]}} & 8'hb0)^
({{8{data_in[166][0]}} & 8'h35)^
({{8{data_in[166][1]}} & 8'h6a)^
({{8{data_in[166][2]}} & 8'hd4)^
({{8{data_in[166][3]}} & 8'h83)^
({{8{data_in[166][4]}} & 8'h2d)^
({{8{data_in[166][5]}} & 8'h5a)^
({{8{data_in[166][6]}} & 8'hb4)^
({{8{data_in[166][7]}} & 8'h43)^
({{8{data_in[167][0]}} & 8'h79)^
({{8{data_in[167][1]}} & 8'hf2)^
({{8{data_in[167][2]}} & 8'hcf)^
({{8{data_in[167][3]}} & 8'hb5)^
({{8{data_in[167][4]}} & 8'h41)^
({{8{data_in[167][5]}} & 8'h82)^
({{8{data_in[167][6]}} & 8'h2f)^
({{8{data_in[167][7]}} & 8'h5e)^
({{8{data_in[168][0]}} & 8'h5e)^
({{8{data_in[168][1]}} & 8'hbc)^
({{8{data_in[168][2]}} & 8'h53)^
({{8{data_in[168][3]}} & 8'ha6)^
({{8{data_in[168][4]}} & 8'h67)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[168][5]}} & 8'hce)^
({{8{data_in[168][6]}} & 8'hb7)^
({{8{data_in[168][7]}} & 8'h45)^
({{8{data_in[169][0]}} & 8'h3d)^
({{8{data_in[169][1]}} & 8'h7a)^
({{8{data_in[169][2]}} & 8'hf4)^
({{8{data_in[169][3]}} & 8'hc3)^
({{8{data_in[169][4]}} & 8'had)^
({{8{data_in[169][5]}} & 8'h71)^
({{8{data_in[169][6]}} & 8'he2)^
({{8{data_in[169][7]}} & 8'hef)^
({{8{data_in[170][0]}} & 8'h1)^
({{8{data_in[170][1]}} & 8'h2)^
({{8{data_in[170][2]}} & 8'h4)^
({{8{data_in[170][3]}} & 8'h8)^
({{8{data_in[170][4]}} & 8'h10)^
({{8{data_in[170][5]}} & 8'h20)^
({{8{data_in[170][6]}} & 8'h40)^
({{8{data_in[170][7]}} & 8'h80)^
({{8{data_in[171][0]}} & 8'h94)^
({{8{data_in[171][1]}} & 8'h3)^
({{8{data_in[171][2]}} & 8'h6)^
({{8{data_in[171][3]}} & 8'hc)^
({{8{data_in[171][4]}} & 8'h18)^
({{8{data_in[171][5]}} & 8'h30)^
({{8{data_in[171][6]}} & 8'h60)^
({{8{data_in[171][7]}} & 8'hc0)^
({{8{data_in[172][0]}} & 8'h4b)^
({{8{data_in[172][1]}} & 8'h96)^
({{8{data_in[172][2]}} & 8'h7)^
({{8{data_in[172][3]}} & 8'he)^
({{8{data_in[172][4]}} & 8'h1c)^
({{8{data_in[172][5]}} & 8'h38)^
({{8{data_in[172][6]}} & 8'h70)^
({{8{data_in[172][7]}} & 8'he0)^
({{8{data_in[173][0]}} & 8'h26)^
({{8{data_in[173][1]}} & 8'h4c)^
({{8{data_in[173][2]}} & 8'h98)^
({{8{data_in[173][3]}} & 8'h1b)^
({{8{data_in[173][4]}} & 8'h36)^
({{8{data_in[173][5]}} & 8'h6c)^
({{8{data_in[173][6]}} & 8'hd8)^
({{8{data_in[173][7]}} & 8'h9b)^
({{8{data_in[174][0]}} & 8'h58)^
({{8{data_in[174][1]}} & 8'hb0)^
({{8{data_in[174][2]}} & 8'h4b)^
({{8{data_in[174][3]}} & 8'h96)^
({{8{data_in[174][4]}} & 8'h7)^
({{8{data_in[174][5]}} & 8'he)^
({{8{data_in[174][6]}} & 8'h1c)^
({{8{data_in[174][7]}} & 8'h38)^
({{8{data_in[175][0]}} & 8'hbe)^
({{8{data_in[175][1]}} & 8'h57)^
({{8{data_in[175][2]}} & 8'hae)^
({{8{data_in[175][3]}} & 8'h77)^
({{8{data_in[175][4]}} & 8'hee)^
({{8{data_in[175][5]}} & 8'hf7)^
({{8{data_in[175][6]}} & 8'hc5)^
({{8{data_in[175][7]}} & 8'ha1)^
({{8{data_in[176][0]}} & 8'h94)^
({{8{data_in[176][1]}} & 8'h3)^
({{8{data_in[176][2]}} & 8'h6)^
({{8{data_in[176][3]}} & 8'hc)^
({{8{data_in[176][4]}} & 8'h18)^
({{8{data_in[176][5]}} & 8'h30)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[176][6]}} & 8'h60)^
({{8{data_in[176][7]}} & 8'hc0)^
({{8{data_in[177][0]}} & 8'h3b)^
({{8{data_in[177][1]}} & 8'h76)^
({{8{data_in[177][2]}} & 8'hec)^
({{8{data_in[177][3]}} & 8'hf3)^
({{8{data_in[177][4]}} & 8'hcd)^
({{8{data_in[177][5]}} & 8'hb1)^
({{8{data_in[177][6]}} & 8'h49)^
({{8{data_in[177][7]}} & 8'h92)^
({{8{data_in[178][0]}} & 8'h39)^
({{8{data_in[178][1]}} & 8'h72)^
({{8{data_in[178][2]}} & 8'he4)^
({{8{data_in[178][3]}} & 8'he3)^
({{8{data_in[178][4]}} & 8'hed)^
({{8{data_in[178][5]}} & 8'hf1)^
({{8{data_in[178][6]}} & 8'hc9)^
({{8{data_in[178][7]}} & 8'hb9)^
({{8{data_in[179][0]}} & 8'h99)^
({{8{data_in[179][1]}} & 8'h19)^
({{8{data_in[179][2]}} & 8'h32)^
({{8{data_in[179][3]}} & 8'h64)^
({{8{data_in[179][4]}} & 8'hc8)^
({{8{data_in[179][5]}} & 8'hbb)^
({{8{data_in[179][6]}} & 8'h5d)^
({{8{data_in[179][7]}} & 8'hba)^
({{8{data_in[180][0]}} & 8'hb)^
({{8{data_in[180][1]}} & 8'h16)^
({{8{data_in[180][2]}} & 8'h2c)^
({{8{data_in[180][3]}} & 8'h58)^
({{8{data_in[180][4]}} & 8'hb0)^
({{8{data_in[180][5]}} & 8'h4b)^
({{8{data_in[180][6]}} & 8'h96)^
({{8{data_in[180][7]}} & 8'h7)^
({{8{data_in[181][0]}} & 8'h1)^
({{8{data_in[181][1]}} & 8'h2)^
({{8{data_in[181][2]}} & 8'h4)^
({{8{data_in[181][3]}} & 8'h8)^
({{8{data_in[181][4]}} & 8'h10)^
({{8{data_in[181][5]}} & 8'h20)^
({{8{data_in[181][6]}} & 8'h40)^
({{8{data_in[181][7]}} & 8'h80)^
({{8{data_in[182][0]}} & 8'h2a)^
({{8{data_in[182][1]}} & 8'h54)^
({{8{data_in[182][2]}} & 8'ha8)^
({{8{data_in[182][3]}} & 8'h7b)^
({{8{data_in[182][4]}} & 8'hf6)^
({{8{data_in[182][5]}} & 8'hc7)^
({{8{data_in[182][6]}} & 8'ha5)^
({{8{data_in[182][7]}} & 8'h61)^
({{8{data_in[183][0]}} & 8'ha0)^
({{8{data_in[183][1]}} & 8'h6b)^
({{8{data_in[183][2]}} & 8'hd6)^
({{8{data_in[183][3]}} & 8'h87)^
({{8{data_in[183][4]}} & 8'h25)^
({{8{data_in[183][5]}} & 8'h4a)^
({{8{data_in[183][6]}} & 8'h94)^
({{8{data_in[183][7]}} & 8'h3)^
({{8{data_in[184][0]}} & 8'h37)^
({{8{data_in[184][1]}} & 8'h6e)^
({{8{data_in[184][2]}} & 8'hdc)^
({{8{data_in[184][3]}} & 8'h93)^
({{8{data_in[184][4]}} & 8'hd)^
({{8{data_in[184][5]}} & 8'h1a)^
({{8{data_in[184][6]}} & 8'h34)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[184][7]}} & 8'h68)^
({{8{data_in[185][0]}} & 8'he1)^
({{8{data_in[185][1]}} & 8'he9)^
({{8{data_in[185][2]}} & 8'hf9)^
({{8{data_in[185][3]}} & 8'hd9)^
({{8{data_in[185][4]}} & 8'h99)^
({{8{data_in[185][5]}} & 8'h19)^
({{8{data_in[185][6]}} & 8'h32)^
({{8{data_in[185][7]}} & 8'h64)^
({{8{data_in[186][0]}} & 8'hd5)^
({{8{data_in[186][1]}} & 8'h81)^
({{8{data_in[186][2]}} & 8'h29)^
({{8{data_in[186][3]}} & 8'h52)^
({{8{data_in[186][4]}} & 8'ha4)^
({{8{data_in[186][5]}} & 8'h63)^
({{8{data_in[186][6]}} & 8'hc6)^
({{8{data_in[186][7]}} & 8'ha7)^
({{8{data_in[187][0]}} & 8'h56)^
({{8{data_in[187][1]}} & 8'hac)^
({{8{data_in[187][2]}} & 8'h73)^
({{8{data_in[187][3]}} & 8'he6)^
({{8{data_in[187][4]}} & 8'he7)^
({{8{data_in[187][5]}} & 8'he5)^
({{8{data_in[187][6]}} & 8'he1)^
({{8{data_in[187][7]}} & 8'he9)^
({{8{data_in[188][0]}} & 8'h22)^
({{8{data_in[188][1]}} & 8'h44)^
({{8{data_in[188][2]}} & 8'h88)^
({{8{data_in[188][3]}} & 8'h3b)^
({{8{data_in[188][4]}} & 8'h76)^
({{8{data_in[188][5]}} & 8'hec)^
({{8{data_in[188][6]}} & 8'hf3)^
({{8{data_in[188][7]}} & 8'hcd)^
({{8{data_in[189][0]}} & 8'he1)^
({{8{data_in[189][1]}} & 8'he9)^
({{8{data_in[189][2]}} & 8'hf9)^
({{8{data_in[189][3]}} & 8'hd9)^
({{8{data_in[189][4]}} & 8'h99)^
({{8{data_in[189][5]}} & 8'h19)^
({{8{data_in[189][6]}} & 8'h32)^
({{8{data_in[189][7]}} & 8'h64)^
({{8{data_in[190][0]}} & 8'hcf)^
({{8{data_in[190][1]}} & 8'hb5)^
({{8{data_in[190][2]}} & 8'h41)^
({{8{data_in[190][3]}} & 8'h82)^
({{8{data_in[190][4]}} & 8'h2f)^
({{8{data_in[190][5]}} & 8'h5e)^
({{8{data_in[190][6]}} & 8'hbc)^
({{8{data_in[190][7]}} & 8'h53)^
({{8{data_in[191][0]}} & 8'h28)^
({{8{data_in[191][1]}} & 8'h50)^
({{8{data_in[191][2]}} & 8'ha0)^
({{8{data_in[191][3]}} & 8'h6b)^
({{8{data_in[191][4]}} & 8'hd6)^
({{8{data_in[191][5]}} & 8'h87)^
({{8{data_in[191][6]}} & 8'h25)^
({{8{data_in[191][7]}} & 8'h4a)^
({{8{data_in[192][0]}} & 8'hbe)^
({{8{data_in[192][1]}} & 8'h57)^
({{8{data_in[192][2]}} & 8'hae)^
({{8{data_in[192][3]}} & 8'h77)^
({{8{data_in[192][4]}} & 8'hee)^
({{8{data_in[192][5]}} & 8'hf7)^
({{8{data_in[192][6]}} & 8'hc5)^
({{8{data_in[192][7]}} & 8'ha1)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[193][0]}} & 8'h96)^
({{8{data_in[193][1]}} & 8'h7)^
({{8{data_in[193][2]}} & 8'he)^
({{8{data_in[193][3]}} & 8'h1c)^
({{8{data_in[193][4]}} & 8'h38)^
({{8{data_in[193][5]}} & 8'h70)^
({{8{data_in[193][6]}} & 8'he0)^
({{8{data_in[193][7]}} & 8'heb)^
({{8{data_in[194][0]}} & 8'h9)^
({{8{data_in[194][1]}} & 8'h12)^
({{8{data_in[194][2]}} & 8'h24)^
({{8{data_in[194][3]}} & 8'h48)^
({{8{data_in[194][4]}} & 8'h90)^
({{8{data_in[194][5]}} & 8'hb)^
({{8{data_in[194][6]}} & 8'h16)^
({{8{data_in[194][7]}} & 8'h2c)^
({{8{data_in[195][0]}} & 8'h10)^
({{8{data_in[195][1]}} & 8'h20)^
({{8{data_in[195][2]}} & 8'h40)^
({{8{data_in[195][3]}} & 8'h80)^
({{8{data_in[195][4]}} & 8'h2b)^
({{8{data_in[195][5]}} & 8'h56)^
({{8{data_in[195][6]}} & 8'hac)^
({{8{data_in[195][7]}} & 8'h73)^
({{8{data_in[196][0]}} & 8'hf0)^
({{8{data_in[196][1]}} & 8'hcb)^
({{8{data_in[196][2]}} & 8'hbd)^
({{8{data_in[196][3]}} & 8'h51)^
({{8{data_in[196][4]}} & 8'ha2)^
({{8{data_in[196][5]}} & 8'h6f)^
({{8{data_in[196][6]}} & 8'hde)^
({{8{data_in[196][7]}} & 8'h97)^
({{8{data_in[197][0]}} & 8'hec)^
({{8{data_in[197][1]}} & 8'hf3)^
({{8{data_in[197][2]}} & 8'hcd)^
({{8{data_in[197][3]}} & 8'hb1)^
({{8{data_in[197][4]}} & 8'h49)^
({{8{data_in[197][5]}} & 8'h92)^
({{8{data_in[197][6]}} & 8'hf)^
({{8{data_in[197][7]}} & 8'h1e)^
({{8{data_in[198][0]}} & 8'h83)^
({{8{data_in[198][1]}} & 8'h2d)^
({{8{data_in[198][2]}} & 8'h5a)^
({{8{data_in[198][3]}} & 8'hb4)^
({{8{data_in[198][4]}} & 8'h43)^
({{8{data_in[198][5]}} & 8'h86)^
({{8{data_in[198][6]}} & 8'h27)^
({{8{data_in[198][7]}} & 8'h4e)^
({{8{data_in[199][0]}} & 8'h96)^
({{8{data_in[199][1]}} & 8'h7)^
({{8{data_in[199][2]}} & 8'he)^
({{8{data_in[199][3]}} & 8'h1c)^
({{8{data_in[199][4]}} & 8'h38)^
({{8{data_in[199][5]}} & 8'h70)^
({{8{data_in[199][6]}} & 8'he0)^
({{8{data_in[199][7]}} & 8'heb)^
({{8{data_in[200][0]}} & 8'h9a)^
({{8{data_in[200][1]}} & 8'h1f)^
({{8{data_in[200][2]}} & 8'h3e)^
({{8{data_in[200][3]}} & 8'h7c)^
({{8{data_in[200][4]}} & 8'hf8)^
({{8{data_in[200][5]}} & 8'hdb)^
({{8{data_in[200][6]}} & 8'h9d)^
({{8{data_in[200][7]}} & 8'h11)^
({{8{data_in[201][0]}} & 8'h50)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[201][1]}} & 8'ha0)^
({{8{data_in[201][2]}} & 8'h6b)^
({{8{data_in[201][3]}} & 8'hd6)^
({{8{data_in[201][4]}} & 8'h87)^
({{8{data_in[201][5]}} & 8'h25)^
({{8{data_in[201][6]}} & 8'h4a)^
({{8{data_in[201][7]}} & 8'h94)^
({{8{data_in[202][0]}} & 8'h4f)^
({{8{data_in[202][1]}} & 8'h9e)^
({{8{data_in[202][2]}} & 8'h17)^
({{8{data_in[202][3]}} & 8'h2e)^
({{8{data_in[202][4]}} & 8'h5c)^
({{8{data_in[202][5]}} & 8'hb8)^
({{8{data_in[202][6]}} & 8'h5b)^
({{8{data_in[202][7]}} & 8'hb6)^
({{8{data_in[203][0]}} & 8'hb9)^
({{8{data_in[203][1]}} & 8'h59)^
({{8{data_in[203][2]}} & 8'hb2)^
({{8{data_in[203][3]}} & 8'h4f)^
({{8{data_in[203][4]}} & 8'h9e)^
({{8{data_in[203][5]}} & 8'h17)^
({{8{data_in[203][6]}} & 8'h2e)^
({{8{data_in[203][7]}} & 8'h5c)^
({{8{data_in[204][0]}} & 8'h19)^
({{8{data_in[204][1]}} & 8'h32)^
({{8{data_in[204][2]}} & 8'h64)^
({{8{data_in[204][3]}} & 8'hc8)^
({{8{data_in[204][4]}} & 8'hbb)^
({{8{data_in[204][5]}} & 8'h5d)^
({{8{data_in[204][6]}} & 8'hba)^
({{8{data_in[204][7]}} & 8'h5f)^
({{8{data_in[205][0]}} & 8'hc0)^
({{8{data_in[205][1]}} & 8'hab)^
({{8{data_in[205][2]}} & 8'h7d)^
({{8{data_in[205][3]}} & 8'hfa)^
({{8{data_in[205][4]}} & 8'hdf)^
({{8{data_in[205][5]}} & 8'h95)^
({{8{data_in[205][6]}} & 8'h1)^
({{8{data_in[205][7]}} & 8'h2)^
({{8{data_in[206][0]}} & 8'hd2)^
({{8{data_in[206][1]}} & 8'h8f)^
({{8{data_in[206][2]}} & 8'h35)^
({{8{data_in[206][3]}} & 8'h6a)^
({{8{data_in[206][4]}} & 8'hd4)^
({{8{data_in[206][5]}} & 8'h83)^
({{8{data_in[206][6]}} & 8'h2d)^
({{8{data_in[206][7]}} & 8'h5a)^
({{8{data_in[207][0]}} & 8'h96)^
({{8{data_in[207][1]}} & 8'h7)^
({{8{data_in[207][2]}} & 8'he)^
({{8{data_in[207][3]}} & 8'h1c)^
({{8{data_in[207][4]}} & 8'h38)^
({{8{data_in[207][5]}} & 8'h70)^
({{8{data_in[207][6]}} & 8'he0)^
({{8{data_in[207][7]}} & 8'heb)^
({{8{data_in[208][0]}} & 8'hc6)^
({{8{data_in[208][1]}} & 8'ha7)^
({{8{data_in[208][2]}} & 8'h65)^
({{8{data_in[208][3]}} & 8'hca)^
({{8{data_in[208][4]}} & 8'hbf)^
({{8{data_in[208][5]}} & 8'h55)^
({{8{data_in[208][6]}} & 8'haa)^
({{8{data_in[208][7]}} & 8'h7f)^
({{8{data_in[209][0]}} & 8'h15)^
({{8{data_in[209][1]}} & 8'h2a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[209][2]}} & 8'h54)^
({{8{data_in[209][3]}} & 8'ha8)^
({{8{data_in[209][4]}} & 8'h7b)^
({{8{data_in[209][5]}} & 8'hf6)^
({{8{data_in[209][6]}} & 8'hc7)^
({{8{data_in[209][7]}} & 8'ha5)^
({{8{data_in[210][0]}} & 8'h70)^
({{8{data_in[210][1]}} & 8'he0)^
({{8{data_in[210][2]}} & 8'heb)^
({{8{data_in[210][3]}} & 8'hfd)^
({{8{data_in[210][4]}} & 8'hd1)^
({{8{data_in[210][5]}} & 8'h89)^
({{8{data_in[210][6]}} & 8'h39)^
({{8{data_in[210][7]}} & 8'h72)^
({{8{data_in[211][0]}} & 8'hed)^
({{8{data_in[211][1]}} & 8'hf1)^
({{8{data_in[211][2]}} & 8'hc9)^
({{8{data_in[211][3]}} & 8'hb9)^
({{8{data_in[211][4]}} & 8'h59)^
({{8{data_in[211][5]}} & 8'hb2)^
({{8{data_in[211][6]}} & 8'h4f)^
({{8{data_in[211][7]}} & 8'h9e)^
({{8{data_in[212][0]}} & 8'h66)^
({{8{data_in[212][1]}} & 8'hcc)^
({{8{data_in[212][2]}} & 8'hb3)^
({{8{data_in[212][3]}} & 8'h4d)^
({{8{data_in[212][4]}} & 8'h9a)^
({{8{data_in[212][5]}} & 8'h1f)^
({{8{data_in[212][6]}} & 8'h3e)^
({{8{data_in[212][7]}} & 8'h7c)^
({{8{data_in[213][0]}} & 8'hf3)^
({{8{data_in[213][1]}} & 8'hcd)^
({{8{data_in[213][2]}} & 8'hb1)^
({{8{data_in[213][3]}} & 8'h49)^
({{8{data_in[213][4]}} & 8'h92)^
({{8{data_in[213][5]}} & 8'hf)^
({{8{data_in[213][6]}} & 8'h1e)^
({{8{data_in[213][7]}} & 8'h3c)^
({{8{data_in[214][0]}} & 8'hb6)^
({{8{data_in[214][1]}} & 8'h47)^
({{8{data_in[214][2]}} & 8'h8e)^
({{8{data_in[214][3]}} & 8'h37)^
({{8{data_in[214][4]}} & 8'h6e)^
({{8{data_in[214][5]}} & 8'hdc)^
({{8{data_in[214][6]}} & 8'h93)^
({{8{data_in[214][7]}} & 8'hd)^
({{8{data_in[215][0]}} & 8'h13)^
({{8{data_in[215][1]}} & 8'h26)^
({{8{data_in[215][2]}} & 8'h4c)^
({{8{data_in[215][3]}} & 8'h98)^
({{8{data_in[215][4]}} & 8'h1b)^
({{8{data_in[215][5]}} & 8'h36)^
({{8{data_in[215][6]}} & 8'h6c)^
({{8{data_in[215][7]}} & 8'hd8)^
({{8{data_in[216][0]}} & 8'hc5)^
({{8{data_in[216][1]}} & 8'ha1)^
({{8{data_in[216][2]}} & 8'h69)^
({{8{data_in[216][3]}} & 8'hd2)^
({{8{data_in[216][4]}} & 8'h8f)^
({{8{data_in[216][5]}} & 8'h35)^
({{8{data_in[216][6]}} & 8'h6a)^
({{8{data_in[216][7]}} & 8'hd4)^
({{8{data_in[217][0]}} & 8'hd3)^
({{8{data_in[217][1]}} & 8'h8d)^
({{8{data_in[217][2]}} & 8'h31)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[217][3]}} & 8'h62)^
({{8{data_in[217][4]}} & 8'hc4)^
({{8{data_in[217][5]}} & 8'ha3)^
({{8{data_in[217][6]}} & 8'h6d)^
({{8{data_in[217][7]}} & 8'hda)^
({{8{data_in[218][0]}} & 8'he2)^
({{8{data_in[218][1]}} & 8'hef)^
({{8{data_in[218][2]}} & 8'hf5)^
({{8{data_in[218][3]}} & 8'hc1)^
({{8{data_in[218][4]}} & 8'ha9)^
({{8{data_in[218][5]}} & 8'h79)^
({{8{data_in[218][6]}} & 8'hf2)^
({{8{data_in[218][7]}} & 8'hcf)^
({{8{data_in[219][0]}} & 8'h46)^
({{8{data_in[219][1]}} & 8'h8c)^
({{8{data_in[219][2]}} & 8'h33)^
({{8{data_in[219][3]}} & 8'h66)^
({{8{data_in[219][4]}} & 8'hcc)^
({{8{data_in[219][5]}} & 8'hb3)^
({{8{data_in[219][6]}} & 8'h4d)^
({{8{data_in[219][7]}} & 8'h9a)^
({{8{data_in[220][0]}} & 8'h3d)^
({{8{data_in[220][1]}} & 8'h7a)^
({{8{data_in[220][2]}} & 8'hf4)^
({{8{data_in[220][3]}} & 8'hc3)^
({{8{data_in[220][4]}} & 8'had)^
({{8{data_in[220][5]}} & 8'h71)^
({{8{data_in[220][6]}} & 8'he2)^
({{8{data_in[220][7]}} & 8'hef)^
({{8{data_in[221][0]}} & 8'hc5)^
({{8{data_in[221][1]}} & 8'ha1)^
({{8{data_in[221][2]}} & 8'h69)^
({{8{data_in[221][3]}} & 8'hd2)^
({{8{data_in[221][4]}} & 8'h8f)^
({{8{data_in[221][5]}} & 8'h35)^
({{8{data_in[221][6]}} & 8'h6a)^
({{8{data_in[221][7]}} & 8'hd4)^
({{8{data_in[222][0]}} & 8'h43)^
({{8{data_in[222][1]}} & 8'h86)^
({{8{data_in[222][2]}} & 8'h27)^
({{8{data_in[222][3]}} & 8'h4e)^
({{8{data_in[222][4]}} & 8'h9c)^
({{8{data_in[222][5]}} & 8'h13)^
({{8{data_in[222][6]}} & 8'h26)^
({{8{data_in[222][7]}} & 8'h4c)^
({{8{data_in[223][0]}} & 8'ha3)^
({{8{data_in[223][1]}} & 8'h6d)^
({{8{data_in[223][2]}} & 8'hda)^
({{8{data_in[223][3]}} & 8'h9f)^
({{8{data_in[223][4]}} & 8'h15)^
({{8{data_in[223][5]}} & 8'h2a)^
({{8{data_in[223][6]}} & 8'h54)^
({{8{data_in[223][7]}} & 8'ha8)^
({{8{data_in[224][0]}} & 8'h55)^
({{8{data_in[224][1]}} & 8'haa)^
({{8{data_in[224][2]}} & 8'h7f)^
({{8{data_in[224][3]}} & 8'hfe)^
({{8{data_in[224][4]}} & 8'hd7)^
({{8{data_in[224][5]}} & 8'h85)^
({{8{data_in[224][6]}} & 8'h21)^
({{8{data_in[224][7]}} & 8'h42)^
({{8{data_in[225][0]}} & 8'h5e)^
({{8{data_in[225][1]}} & 8'hbc)^
({{8{data_in[225][2]}} & 8'h53)^
({{8{data_in[225][3]}} & 8'ha6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[225][4]}} & 8'h67)^
({{8{data_in[225][5]}} & 8'hce)^
({{8{data_in[225][6]}} & 8'hb7)^
({{8{data_in[225][7]}} & 8'h45)^
({{8{data_in[226][0]}} & 8'h9b)^
({{8{data_in[226][1]}} & 8'h1d)^
({{8{data_in[226][2]}} & 8'h3a)^
({{8{data_in[226][3]}} & 8'h74)^
({{8{data_in[226][4]}} & 8'he8)^
({{8{data_in[226][5]}} & 8'hfb)^
({{8{data_in[226][6]}} & 8'hdd)^
({{8{data_in[226][7]}} & 8'h91)^
({{8{data_in[227][0]}} & 8'hba)^
({{8{data_in[227][1]}} & 8'h5f)^
({{8{data_in[227][2]}} & 8'hbe)^
({{8{data_in[227][3]}} & 8'h57)^
({{8{data_in[227][4]}} & 8'hae)^
({{8{data_in[227][5]}} & 8'h77)^
({{8{data_in[227][6]}} & 8'hee)^
({{8{data_in[227][7]}} & 8'hf7)^
({{8{data_in[228][0]}} & 8'haa)^
({{8{data_in[228][1]}} & 8'h7f)^
({{8{data_in[228][2]}} & 8'hfe)^
({{8{data_in[228][3]}} & 8'hd7)^
({{8{data_in[228][4]}} & 8'h85)^
({{8{data_in[228][5]}} & 8'h21)^
({{8{data_in[228][6]}} & 8'h42)^
({{8{data_in[228][7]}} & 8'h84)^
({{8{data_in[229][0]}} & 8'h7)^
({{8{data_in[229][1]}} & 8'he)^
({{8{data_in[229][2]}} & 8'h1c)^
({{8{data_in[229][3]}} & 8'h38)^
({{8{data_in[229][4]}} & 8'h70)^
({{8{data_in[229][5]}} & 8'he0)^
({{8{data_in[229][6]}} & 8'heb)^
({{8{data_in[229][7]}} & 8'hfd)^
({{8{data_in[230][0]}} & 8'h4f)^
({{8{data_in[230][1]}} & 8'h9e)^
({{8{data_in[230][2]}} & 8'h17)^
({{8{data_in[230][3]}} & 8'h2e)^
({{8{data_in[230][4]}} & 8'h5c)^
({{8{data_in[230][5]}} & 8'hb8)^
({{8{data_in[230][6]}} & 8'h5b)^
({{8{data_in[230][7]}} & 8'hb6)^
({{8{data_in[231][0]}} & 8'h6c)^
({{8{data_in[231][1]}} & 8'hd8)^
({{8{data_in[231][2]}} & 8'h9b)^
({{8{data_in[231][3]}} & 8'h1d)^
({{8{data_in[231][4]}} & 8'h3a)^
({{8{data_in[231][5]}} & 8'h74)^
({{8{data_in[231][6]}} & 8'he8)^
({{8{data_in[231][7]}} & 8'hfb)^
({{8{data_in[232][0]}} & 8'h89)^
({{8{data_in[232][1]}} & 8'h39)^
({{8{data_in[232][2]}} & 8'h72)^
({{8{data_in[232][3]}} & 8'he4)^
({{8{data_in[232][4]}} & 8'he3)^
({{8{data_in[232][5]}} & 8'hed)^
({{8{data_in[232][6]}} & 8'hf1)^
({{8{data_in[232][7]}} & 8'hc9)^
({{8{data_in[233][0]}} & 8'h40)^
({{8{data_in[233][1]}} & 8'h80)^
({{8{data_in[233][2]}} & 8'h2b)^
({{8{data_in[233][3]}} & 8'h56)^
({{8{data_in[233][4]}} & 8'hac)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[233][5]}} & 8'h73)^
({{8{data_in[233][6]}} & 8'he6)^
({{8{data_in[233][7]}} & 8'he7)^
({{8{data_in[234][0]}} & 8'h51)^
({{8{data_in[234][1]}} & 8'ha2)^
({{8{data_in[234][2]}} & 8'h6f)^
({{8{data_in[234][3]}} & 8'hde)^
({{8{data_in[234][4]}} & 8'h97)^
({{8{data_in[234][5]}} & 8'h5)^
({{8{data_in[234][6]}} & 8'ha)^
({{8{data_in[234][7]}} & 8'h14)^
({{8{data_in[235][0]}} & 8'hb4)^
({{8{data_in[235][1]}} & 8'h43)^
({{8{data_in[235][2]}} & 8'h86)^
({{8{data_in[235][3]}} & 8'h27)^
({{8{data_in[235][4]}} & 8'h4e)^
({{8{data_in[235][5]}} & 8'h9c)^
({{8{data_in[235][6]}} & 8'h13)^
({{8{data_in[235][7]}} & 8'h26)^
({{8{data_in[236][0]}} & 8'hef)^
({{8{data_in[236][1]}} & 8'hf5)^
({{8{data_in[236][2]}} & 8'hc1)^
({{8{data_in[236][3]}} & 8'ha9)^
({{8{data_in[236][4]}} & 8'h79)^
({{8{data_in[236][5]}} & 8'hf2)^
({{8{data_in[236][6]}} & 8'hcf)^
({{8{data_in[236][7]}} & 8'hb5)^
({{8{data_in[237][0]}} & 8'h30)^
({{8{data_in[237][1]}} & 8'h60)^
({{8{data_in[237][2]}} & 8'hc0)^
({{8{data_in[237][3]}} & 8'hab)^
({{8{data_in[237][4]}} & 8'h7d)^
({{8{data_in[237][5]}} & 8'hfa)^
({{8{data_in[237][6]}} & 8'hdf)^
({{8{data_in[237][7]}} & 8'h95)^
({{8{data_in[238][0]}} & 8'hc2)^
({{8{data_in[238][1]}} & 8'haf)^
({{8{data_in[238][2]}} & 8'h75)^
({{8{data_in[238][3]}} & 8'hea)^
({{8{data_in[238][4]}} & 8'hff)^
({{8{data_in[238][5]}} & 8'hd5)^
({{8{data_in[238][6]}} & 8'h81)^
({{8{data_in[238][7]}} & 8'h29)^
({{8{data_in[239][0]}} & 8'hbe)^
({{8{data_in[239][1]}} & 8'h57)^
({{8{data_in[239][2]}} & 8'hae)^
({{8{data_in[239][3]}} & 8'h77)^
({{8{data_in[239][4]}} & 8'hee)^
({{8{data_in[239][5]}} & 8'hf7)^
({{8{data_in[239][6]}} & 8'hc5)^
({{8{data_in[239][7]}} & 8'ha1)^
({{8{data_in[240][0]}} & 8'h51)^
({{8{data_in[240][1]}} & 8'ha2)^
({{8{data_in[240][2]}} & 8'h6f)^
({{8{data_in[240][3]}} & 8'hde)^
({{8{data_in[240][4]}} & 8'h97)^
({{8{data_in[240][5]}} & 8'h5)^
({{8{data_in[240][6]}} & 8'ha)^
({{8{data_in[240][7]}} & 8'h14)^
({{8{data_in[241][0]}} & 8'hd5)^
({{8{data_in[241][1]}} & 8'h81)^
({{8{data_in[241][2]}} & 8'h29)^
({{8{data_in[241][3]}} & 8'h52)^
({{8{data_in[241][4]}} & 8'ha4)^
({{8{data_in[241][5]}} & 8'h63)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[241][6]}} & 8'hc6)^
({{8{data_in[241][7]}} & 8'ha7);
data_out[248] = ({{8{data_in[0][0]}} & 8'hb9)^
({{8{data_in[0][1]}} & 8'h59)^
({{8{data_in[0][2]}} & 8'hb2)^
({{8{data_in[0][3]}} & 8'h4f)^
({{8{data_in[0][4]}} & 8'h9e)^
({{8{data_in[0][5]}} & 8'h17)^
({{8{data_in[0][6]}} & 8'h2e)^
({{8{data_in[0][7]}} & 8'h5c)^
({{8{data_in[1][0]}} & 8'h43)^
({{8{data_in[1][1]}} & 8'h86)^
({{8{data_in[1][2]}} & 8'h27)^
({{8{data_in[1][3]}} & 8'h4e)^
({{8{data_in[1][4]}} & 8'h9c)^
({{8{data_in[1][5]}} & 8'h13)^
({{8{data_in[1][6]}} & 8'h26)^
({{8{data_in[1][7]}} & 8'h4c)^
({{8{data_in[2][0]}} & 8'he2)^
({{8{data_in[2][1]}} & 8'hef)^
({{8{data_in[2][2]}} & 8'hf5)^
({{8{data_in[2][3]}} & 8'hc1)^
({{8{data_in[2][4]}} & 8'ha9)^
({{8{data_in[2][5]}} & 8'h79)^
({{8{data_in[2][6]}} & 8'hf2)^
({{8{data_in[2][7]}} & 8'hcf)^
({{8{data_in[3][0]}} & 8'hbe)^
({{8{data_in[3][1]}} & 8'h57)^
({{8{data_in[3][2]}} & 8'hae)^
({{8{data_in[3][3]}} & 8'h77)^
({{8{data_in[3][4]}} & 8'hee)^
({{8{data_in[3][5]}} & 8'hf7)^
({{8{data_in[3][6]}} & 8'hc5)^
({{8{data_in[3][7]}} & 8'ha1)^
({{8{data_in[4][0]}} & 8'hb1)^
({{8{data_in[4][1]}} & 8'h49)^
({{8{data_in[4][2]}} & 8'h92)^
({{8{data_in[4][3]}} & 8'hf)^
({{8{data_in[4][4]}} & 8'h1e)^
({{8{data_in[4][5]}} & 8'h3c)^
({{8{data_in[4][6]}} & 8'h78)^
({{8{data_in[4][7]}} & 8'hf0)^
({{8{data_in[5][0]}} & 8'h4)^
({{8{data_in[5][1]}} & 8'h8)^
({{8{data_in[5][2]}} & 8'h10)^
({{8{data_in[5][3]}} & 8'h20)^
({{8{data_in[5][4]}} & 8'h40)^
({{8{data_in[5][5]}} & 8'h80)^
({{8{data_in[5][6]}} & 8'h2b)^
({{8{data_in[5][7]}} & 8'h56)^
({{8{data_in[6][0]}} & 8'hb8)^
({{8{data_in[6][1]}} & 8'h5b)^
({{8{data_in[6][2]}} & 8'hb6)^
({{8{data_in[6][3]}} & 8'h47)^
({{8{data_in[6][4]}} & 8'h8e)^
({{8{data_in[6][5]}} & 8'h37)^
({{8{data_in[6][6]}} & 8'h6e)^
({{8{data_in[6][7]}} & 8'hdc)^
({{8{data_in[7][0]}} & 8'h96)^
({{8{data_in[7][1]}} & 8'h7)^
({{8{data_in[7][2]}} & 8'he)^
({{8{data_in[7][3]}} & 8'h1c)^
({{8{data_in[7][4]}} & 8'h38)^
({{8{data_in[7][5]}} & 8'h70)^
({{8{data_in[7][6]}} & 8'he0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[7][7]}} & 8'heb)^
({{8{data_in[8][0]}} & 8'h3a)^
({{8{data_in[8][1]}} & 8'h74)^
({{8{data_in[8][2]}} & 8'he8)^
({{8{data_in[8][3]}} & 8'hfb)^
({{8{data_in[8][4]}} & 8'hdd)^
({{8{data_in[8][5]}} & 8'h91)^
({{8{data_in[8][6]}} & 8'h9)^
({{8{data_in[8][7]}} & 8'h12)^
({{8{data_in[9][0]}} & 8'h46)^
({{8{data_in[9][1]}} & 8'h8c)^
({{8{data_in[9][2]}} & 8'h33)^
({{8{data_in[9][3]}} & 8'h66)^
({{8{data_in[9][4]}} & 8'hcc)^
({{8{data_in[9][5]}} & 8'hb3)^
({{8{data_in[9][6]}} & 8'h4d)^
({{8{data_in[9][7]}} & 8'h9a)^
({{8{data_in[10][0]}} & 8'h3b)^
({{8{data_in[10][1]}} & 8'h76)^
({{8{data_in[10][2]}} & 8'hec)^
({{8{data_in[10][3]}} & 8'hf3)^
({{8{data_in[10][4]}} & 8'hcd)^
({{8{data_in[10][5]}} & 8'hb1)^
({{8{data_in[10][6]}} & 8'h49)^
({{8{data_in[10][7]}} & 8'h92)^
({{8{data_in[11][0]}} & 8'h7d)^
({{8{data_in[11][1]}} & 8'hfa)^
({{8{data_in[11][2]}} & 8'hdf)^
({{8{data_in[11][3]}} & 8'h95)^
({{8{data_in[11][4]}} & 8'h1)^
({{8{data_in[11][5]}} & 8'h2)^
({{8{data_in[11][6]}} & 8'h4)^
({{8{data_in[11][7]}} & 8'h8)^
({{8{data_in[12][0]}} & 8'h3b)^
({{8{data_in[12][1]}} & 8'h76)^
({{8{data_in[12][2]}} & 8'hec)^
({{8{data_in[12][3]}} & 8'hf3)^
({{8{data_in[12][4]}} & 8'hcd)^
({{8{data_in[12][5]}} & 8'hb1)^
({{8{data_in[12][6]}} & 8'h49)^
({{8{data_in[12][7]}} & 8'h92)^
({{8{data_in[13][0]}} & 8'he3)^
({{8{data_in[13][1]}} & 8'hed)^
({{8{data_in[13][2]}} & 8'hf1)^
({{8{data_in[13][3]}} & 8'hc9)^
({{8{data_in[13][4]}} & 8'hb9)^
({{8{data_in[13][5]}} & 8'h59)^
({{8{data_in[13][6]}} & 8'hb2)^
({{8{data_in[13][7]}} & 8'h4f)^
({{8{data_in[14][0]}} & 8'h18)^
({{8{data_in[14][1]}} & 8'h30)^
({{8{data_in[14][2]}} & 8'h60)^
({{8{data_in[14][3]}} & 8'hc0)^
({{8{data_in[14][4]}} & 8'hab)^
({{8{data_in[14][5]}} & 8'h7d)^
({{8{data_in[14][6]}} & 8'hfa)^
({{8{data_in[14][7]}} & 8'hdf)^
({{8{data_in[15][0]}} & 8'h3a)^
({{8{data_in[15][1]}} & 8'h74)^
({{8{data_in[15][2]}} & 8'he8)^
({{8{data_in[15][3]}} & 8'hfb)^
({{8{data_in[15][4]}} & 8'hdd)^
({{8{data_in[15][5]}} & 8'h91)^
({{8{data_in[15][6]}} & 8'h9)^
({{8{data_in[15][7]}} & 8'h12)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[16][0]}} & 8'h4b)^
({{8{data_in[16][1]}} & 8'h96)^
({{8{data_in[16][2]}} & 8'h7)^
({{8{data_in[16][3]}} & 8'he)^
({{8{data_in[16][4]}} & 8'h1c)^
({{8{data_in[16][5]}} & 8'h38)^
({{8{data_in[16][6]}} & 8'h70)^
({{8{data_in[16][7]}} & 8'he0)^
({{8{data_in[17][0]}} & 8'h97)^
({{8{data_in[17][1]}} & 8'h5)^
({{8{data_in[17][2]}} & 8'ha)^
({{8{data_in[17][3]}} & 8'h14)^
({{8{data_in[17][4]}} & 8'h28)^
({{8{data_in[17][5]}} & 8'h50)^
({{8{data_in[17][6]}} & 8'ha0)^
({{8{data_in[17][7]}} & 8'h6b)^
({{8{data_in[18][0]}} & 8'h65)^
({{8{data_in[18][1]}} & 8'hca)^
({{8{data_in[18][2]}} & 8'hbf)^
({{8{data_in[18][3]}} & 8'h55)^
({{8{data_in[18][4]}} & 8'haa)^
({{8{data_in[18][5]}} & 8'h7f)^
({{8{data_in[18][6]}} & 8'hfe)^
({{8{data_in[18][7]}} & 8'hd7)^
({{8{data_in[19][0]}} & 8'h6f)^
({{8{data_in[19][1]}} & 8'hde)^
({{8{data_in[19][2]}} & 8'h97)^
({{8{data_in[19][3]}} & 8'h5)^
({{8{data_in[19][4]}} & 8'ha)^
({{8{data_in[19][5]}} & 8'h14)^
({{8{data_in[19][6]}} & 8'h28)^
({{8{data_in[19][7]}} & 8'h50)^
({{8{data_in[20][0]}} & 8'hdc)^
({{8{data_in[20][1]}} & 8'h93)^
({{8{data_in[20][2]}} & 8'hd)^
({{8{data_in[20][3]}} & 8'h1a)^
({{8{data_in[20][4]}} & 8'h34)^
({{8{data_in[20][5]}} & 8'h68)^
({{8{data_in[20][6]}} & 8'hd0)^
({{8{data_in[20][7]}} & 8'h8b)^
({{8{data_in[21][0]}} & 8'h33)^
({{8{data_in[21][1]}} & 8'h66)^
({{8{data_in[21][2]}} & 8'hcc)^
({{8{data_in[21][3]}} & 8'hb3)^
({{8{data_in[21][4]}} & 8'h4d)^
({{8{data_in[21][5]}} & 8'h9a)^
({{8{data_in[21][6]}} & 8'h1f)^
({{8{data_in[21][7]}} & 8'h3e)^
({{8{data_in[22][0]}} & 8'hd4)^
({{8{data_in[22][1]}} & 8'h83)^
({{8{data_in[22][2]}} & 8'h2d)^
({{8{data_in[22][3]}} & 8'h5a)^
({{8{data_in[22][4]}} & 8'hb4)^
({{8{data_in[22][5]}} & 8'h43)^
({{8{data_in[22][6]}} & 8'h86)^
({{8{data_in[22][7]}} & 8'h27)^
({{8{data_in[23][0]}} & 8'h53)^
({{8{data_in[23][1]}} & 8'ha6)^
({{8{data_in[23][2]}} & 8'h67)^
({{8{data_in[23][3]}} & 8'hce)^
({{8{data_in[23][4]}} & 8'hb7)^
({{8{data_in[23][5]}} & 8'h45)^
({{8{data_in[23][6]}} & 8'h8a)^
({{8{data_in[23][7]}} & 8'h3f)^
({{8{data_in[24][0]}} & 8'h2f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[24][1]}} & 8'h5e)^
({{8{data_in[24][2]}} & 8'hbc)^
({{8{data_in[24][3]}} & 8'h53)^
({{8{data_in[24][4]}} & 8'ha6)^
({{8{data_in[24][5]}} & 8'h67)^
({{8{data_in[24][6]}} & 8'hce)^
({{8{data_in[24][7]}} & 8'hb7)^
({{8{data_in[25][0]}} & 8'ha1)^
({{8{data_in[25][1]}} & 8'h69)^
({{8{data_in[25][2]}} & 8'hd2)^
({{8{data_in[25][3]}} & 8'h8f)^
({{8{data_in[25][4]}} & 8'h35)^
({{8{data_in[25][5]}} & 8'h6a)^
({{8{data_in[25][6]}} & 8'hd4)^
({{8{data_in[25][7]}} & 8'h83)^
({{8{data_in[26][0]}} & 8'h51)^
({{8{data_in[26][1]}} & 8'ha2)^
({{8{data_in[26][2]}} & 8'h6f)^
({{8{data_in[26][3]}} & 8'hde)^
({{8{data_in[26][4]}} & 8'h97)^
({{8{data_in[26][5]}} & 8'h5)^
({{8{data_in[26][6]}} & 8'ha)^
({{8{data_in[26][7]}} & 8'h14)^
({{8{data_in[27][0]}} & 8'h9)^
({{8{data_in[27][1]}} & 8'h12)^
({{8{data_in[27][2]}} & 8'h24)^
({{8{data_in[27][3]}} & 8'h48)^
({{8{data_in[27][4]}} & 8'h90)^
({{8{data_in[27][5]}} & 8'hb)^
({{8{data_in[27][6]}} & 8'h16)^
({{8{data_in[27][7]}} & 8'h2c)^
({{8{data_in[28][0]}} & 8'h1f)^
({{8{data_in[28][1]}} & 8'h3e)^
({{8{data_in[28][2]}} & 8'h7c)^
({{8{data_in[28][3]}} & 8'hf8)^
({{8{data_in[28][4]}} & 8'hdb)^
({{8{data_in[28][5]}} & 8'h9d)^
({{8{data_in[28][6]}} & 8'h11)^
({{8{data_in[28][7]}} & 8'h22)^
({{8{data_in[29][0]}} & 8'h25)^
({{8{data_in[29][1]}} & 8'h4a)^
({{8{data_in[29][2]}} & 8'h94)^
({{8{data_in[29][3]}} & 8'h3)^
({{8{data_in[29][4]}} & 8'h6)^
({{8{data_in[29][5]}} & 8'hc)^
({{8{data_in[29][6]}} & 8'h18)^
({{8{data_in[29][7]}} & 8'h30)^
({{8{data_in[30][0]}} & 8'h47)^
({{8{data_in[30][1]}} & 8'h8e)^
({{8{data_in[30][2]}} & 8'h37)^
({{8{data_in[30][3]}} & 8'h6e)^
({{8{data_in[30][4]}} & 8'hdc)^
({{8{data_in[30][5]}} & 8'h93)^
({{8{data_in[30][6]}} & 8'hd)^
({{8{data_in[30][7]}} & 8'h1a)^
({{8{data_in[31][0]}} & 8'hd3)^
({{8{data_in[31][1]}} & 8'h8d)^
({{8{data_in[31][2]}} & 8'h31)^
({{8{data_in[31][3]}} & 8'h62)^
({{8{data_in[31][4]}} & 8'hc4)^
({{8{data_in[31][5]}} & 8'ha3)^
({{8{data_in[31][6]}} & 8'h6d)^
({{8{data_in[31][7]}} & 8'hda)^
({{8{data_in[32][0]}} & 8'hf1)^
({{8{data_in[32][1]}} & 8'hc9)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[32][2]}} & 8'hb9)^
({{8{data_in[32][3]}} & 8'h59)^
({{8{data_in[32][4]}} & 8'hb2)^
({{8{data_in[32][5]}} & 8'h4f)^
({{8{data_in[32][6]}} & 8'h9e)^
({{8{data_in[32][7]}} & 8'h17)^
({{8{data_in[33][0]}} & 8'ha6)^
({{8{data_in[33][1]}} & 8'h67)^
({{8{data_in[33][2]}} & 8'hce)^
({{8{data_in[33][3]}} & 8'hb7)^
({{8{data_in[33][4]}} & 8'h45)^
({{8{data_in[33][5]}} & 8'h8a)^
({{8{data_in[33][6]}} & 8'h3f)^
({{8{data_in[33][7]}} & 8'h7e)^
({{8{data_in[34][0]}} & 8'h4d)^
({{8{data_in[34][1]}} & 8'h9a)^
({{8{data_in[34][2]}} & 8'h1f)^
({{8{data_in[34][3]}} & 8'h3e)^
({{8{data_in[34][4]}} & 8'h7c)^
({{8{data_in[34][5]}} & 8'hf8)^
({{8{data_in[34][6]}} & 8'hdb)^
({{8{data_in[34][7]}} & 8'h9d)^
({{8{data_in[35][0]}} & 8'h76)^
({{8{data_in[35][1]}} & 8'hec)^
({{8{data_in[35][2]}} & 8'hf3)^
({{8{data_in[35][3]}} & 8'hcd)^
({{8{data_in[35][4]}} & 8'hb1)^
({{8{data_in[35][5]}} & 8'h49)^
({{8{data_in[35][6]}} & 8'h92)^
({{8{data_in[35][7]}} & 8'hf)^
({{8{data_in[36][0]}} & 8'he8)^
({{8{data_in[36][1]}} & 8'hfb)^
({{8{data_in[36][2]}} & 8'hdd)^
({{8{data_in[36][3]}} & 8'h91)^
({{8{data_in[36][4]}} & 8'h9)^
({{8{data_in[36][5]}} & 8'h12)^
({{8{data_in[36][6]}} & 8'h24)^
({{8{data_in[36][7]}} & 8'h48)^
({{8{data_in[37][0]}} & 8'h7)^
({{8{data_in[37][1]}} & 8'he)^
({{8{data_in[37][2]}} & 8'h1c)^
({{8{data_in[37][3]}} & 8'h38)^
({{8{data_in[37][4]}} & 8'h70)^
({{8{data_in[37][5]}} & 8'he0)^
({{8{data_in[37][6]}} & 8'heb)^
({{8{data_in[37][7]}} & 8'hfd)^
({{8{data_in[38][0]}} & 8'hec)^
({{8{data_in[38][1]}} & 8'hf3)^
({{8{data_in[38][2]}} & 8'hcd)^
({{8{data_in[38][3]}} & 8'hb1)^
({{8{data_in[38][4]}} & 8'h49)^
({{8{data_in[38][5]}} & 8'h92)^
({{8{data_in[38][6]}} & 8'hf)^
({{8{data_in[38][7]}} & 8'h1e)^
({{8{data_in[39][0]}} & 8'h77)^
({{8{data_in[39][1]}} & 8'hee)^
({{8{data_in[39][2]}} & 8'hf7)^
({{8{data_in[39][3]}} & 8'hc5)^
({{8{data_in[39][4]}} & 8'ha1)^
({{8{data_in[39][5]}} & 8'h69)^
({{8{data_in[39][6]}} & 8'hd2)^
({{8{data_in[39][7]}} & 8'h8f)^
({{8{data_in[40][0]}} & 8'h37)^
({{8{data_in[40][1]}} & 8'h6e)^
({{8{data_in[40][2]}} & 8'hdc)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[40][3]}} & 8'h93)^
({{8{data_in[40][4]}} & 8'hd)^
({{8{data_in[40][5]}} & 8'h1a)^
({{8{data_in[40][6]}} & 8'h34)^
({{8{data_in[40][7]}} & 8'h68)^
({{8{data_in[41][0]}} & 8'hc5)^
({{8{data_in[41][1]}} & 8'ha1)^
({{8{data_in[41][2]}} & 8'h69)^
({{8{data_in[41][3]}} & 8'hd2)^
({{8{data_in[41][4]}} & 8'h8f)^
({{8{data_in[41][5]}} & 8'h35)^
({{8{data_in[41][6]}} & 8'h6a)^
({{8{data_in[41][7]}} & 8'hd4)^
({{8{data_in[42][0]}} & 8'h1c)^
({{8{data_in[42][1]}} & 8'h38)^
({{8{data_in[42][2]}} & 8'h70)^
({{8{data_in[42][3]}} & 8'he0)^
({{8{data_in[42][4]}} & 8'heb)^
({{8{data_in[42][5]}} & 8'hfd)^
({{8{data_in[42][6]}} & 8'hd1)^
({{8{data_in[42][7]}} & 8'h89)^
({{8{data_in[43][0]}} & 8'ha1)^
({{8{data_in[43][1]}} & 8'h69)^
({{8{data_in[43][2]}} & 8'hd2)^
({{8{data_in[43][3]}} & 8'h8f)^
({{8{data_in[43][4]}} & 8'h35)^
({{8{data_in[43][5]}} & 8'h6a)^
({{8{data_in[43][6]}} & 8'hd4)^
({{8{data_in[43][7]}} & 8'h83)^
({{8{data_in[44][0]}} & 8'h89)^
({{8{data_in[44][1]}} & 8'h39)^
({{8{data_in[44][2]}} & 8'h72)^
({{8{data_in[44][3]}} & 8'he4)^
({{8{data_in[44][4]}} & 8'he3)^
({{8{data_in[44][5]}} & 8'hed)^
({{8{data_in[44][6]}} & 8'hf1)^
({{8{data_in[44][7]}} & 8'hc9)^
({{8{data_in[45][0]}} & 8'h17)^
({{8{data_in[45][1]}} & 8'h2e)^
({{8{data_in[45][2]}} & 8'h5c)^
({{8{data_in[45][3]}} & 8'hb8)^
({{8{data_in[45][4]}} & 8'h5b)^
({{8{data_in[45][5]}} & 8'hb6)^
({{8{data_in[45][6]}} & 8'h47)^
({{8{data_in[45][7]}} & 8'h8e)^
({{8{data_in[46][0]}} & 8'h47)^
({{8{data_in[46][1]}} & 8'h8e)^
({{8{data_in[46][2]}} & 8'h37)^
({{8{data_in[46][3]}} & 8'h6e)^
({{8{data_in[46][4]}} & 8'hdc)^
({{8{data_in[46][5]}} & 8'h93)^
({{8{data_in[46][6]}} & 8'hd)^
({{8{data_in[46][7]}} & 8'h1a)^
({{8{data_in[47][0]}} & 8'hdb)^
({{8{data_in[47][1]}} & 8'h9d)^
({{8{data_in[47][2]}} & 8'h11)^
({{8{data_in[47][3]}} & 8'h22)^
({{8{data_in[47][4]}} & 8'h44)^
({{8{data_in[47][5]}} & 8'h88)^
({{8{data_in[47][6]}} & 8'h3b)^
({{8{data_in[47][7]}} & 8'h76)^
({{8{data_in[48][0]}} & 8'h16)^
({{8{data_in[48][1]}} & 8'h2c)^
({{8{data_in[48][2]}} & 8'h58)^
({{8{data_in[48][3]}} & 8'hb0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[48][4]}} & 8'h4b)^
({{8{data_in[48][5]}} & 8'h96)^
({{8{data_in[48][6]}} & 8'h7)^
({{8{data_in[48][7]}} & 8'he)^
({{8{data_in[49][0]}} & 8'heb)^
({{8{data_in[49][1]}} & 8'hfd)^
({{8{data_in[49][2]}} & 8'hd1)^
({{8{data_in[49][3]}} & 8'h89)^
({{8{data_in[49][4]}} & 8'h39)^
({{8{data_in[49][5]}} & 8'h72)^
({{8{data_in[49][6]}} & 8'he4)^
({{8{data_in[49][7]}} & 8'he3)^
({{8{data_in[50][0]}} & 8'h32)^
({{8{data_in[50][1]}} & 8'h64)^
({{8{data_in[50][2]}} & 8'hc8)^
({{8{data_in[50][3]}} & 8'hbb)^
({{8{data_in[50][4]}} & 8'h5d)^
({{8{data_in[50][5]}} & 8'hba)^
({{8{data_in[50][6]}} & 8'h5f)^
({{8{data_in[50][7]}} & 8'hbe)^
({{8{data_in[51][0]}} & 8'h98)^
({{8{data_in[51][1]}} & 8'h1b)^
({{8{data_in[51][2]}} & 8'h36)^
({{8{data_in[51][3]}} & 8'h6c)^
({{8{data_in[51][4]}} & 8'hd8)^
({{8{data_in[51][5]}} & 8'h9b)^
({{8{data_in[51][6]}} & 8'h1d)^
({{8{data_in[51][7]}} & 8'h3a)^
({{8{data_in[52][0]}} & 8'h32)^
({{8{data_in[52][1]}} & 8'h64)^
({{8{data_in[52][2]}} & 8'hc8)^
({{8{data_in[52][3]}} & 8'hbb)^
({{8{data_in[52][4]}} & 8'h5d)^
({{8{data_in[52][5]}} & 8'hba)^
({{8{data_in[52][6]}} & 8'h5f)^
({{8{data_in[52][7]}} & 8'hbe)^
({{8{data_in[53][0]}} & 8'h11)^
({{8{data_in[53][1]}} & 8'h22)^
({{8{data_in[53][2]}} & 8'h44)^
({{8{data_in[53][3]}} & 8'h88)^
({{8{data_in[53][4]}} & 8'h3b)^
({{8{data_in[53][5]}} & 8'h76)^
({{8{data_in[53][6]}} & 8'hec)^
({{8{data_in[53][7]}} & 8'hf3)^
({{8{data_in[54][0]}} & 8'h83)^
({{8{data_in[54][1]}} & 8'h2d)^
({{8{data_in[54][2]}} & 8'h5a)^
({{8{data_in[54][3]}} & 8'hb4)^
({{8{data_in[54][4]}} & 8'h43)^
({{8{data_in[54][5]}} & 8'h86)^
({{8{data_in[54][6]}} & 8'h27)^
({{8{data_in[54][7]}} & 8'h4e)^
({{8{data_in[55][0]}} & 8'hb)^
({{8{data_in[55][1]}} & 8'h16)^
({{8{data_in[55][2]}} & 8'h2c)^
({{8{data_in[55][3]}} & 8'h58)^
({{8{data_in[55][4]}} & 8'hb0)^
({{8{data_in[55][5]}} & 8'h4b)^
({{8{data_in[55][6]}} & 8'h96)^
({{8{data_in[55][7]}} & 8'h7)^
({{8{data_in[56][0]}} & 8'hb4)^
({{8{data_in[56][1]}} & 8'h43)^
({{8{data_in[56][2]}} & 8'h86)^
({{8{data_in[56][3]}} & 8'h27)^
({{8{data_in[56][4]}} & 8'h4e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[56][5]}} & 8'h9c)^
({{8{data_in[56][6]}} & 8'h13)^
({{8{data_in[56][7]}} & 8'h26)^
({{8{data_in[57][0]}} & 8'h31)^
({{8{data_in[57][1]}} & 8'h62)^
({{8{data_in[57][2]}} & 8'hc4)^
({{8{data_in[57][3]}} & 8'ha3)^
({{8{data_in[57][4]}} & 8'h6d)^
({{8{data_in[57][5]}} & 8'hda)^
({{8{data_in[57][6]}} & 8'h9f)^
({{8{data_in[57][7]}} & 8'h15)^
({{8{data_in[58][0]}} & 8'hb1)^
({{8{data_in[58][1]}} & 8'h49)^
({{8{data_in[58][2]}} & 8'h92)^
({{8{data_in[58][3]}} & 8'hf)^
({{8{data_in[58][4]}} & 8'h1e)^
({{8{data_in[58][5]}} & 8'h3c)^
({{8{data_in[58][6]}} & 8'h78)^
({{8{data_in[58][7]}} & 8'hf0)^
({{8{data_in[59][0]}} & 8'he0)^
({{8{data_in[59][1]}} & 8'heb)^
({{8{data_in[59][2]}} & 8'hfd)^
({{8{data_in[59][3]}} & 8'hd1)^
({{8{data_in[59][4]}} & 8'h89)^
({{8{data_in[59][5]}} & 8'h39)^
({{8{data_in[59][6]}} & 8'h72)^
({{8{data_in[59][7]}} & 8'he4)^
({{8{data_in[60][0]}} & 8'h6d)^
({{8{data_in[60][1]}} & 8'hda)^
({{8{data_in[60][2]}} & 8'h9f)^
({{8{data_in[60][3]}} & 8'h15)^
({{8{data_in[60][4]}} & 8'h2a)^
({{8{data_in[60][5]}} & 8'h54)^
({{8{data_in[60][6]}} & 8'ha8)^
({{8{data_in[60][7]}} & 8'h7b)^
({{8{data_in[61][0]}} & 8'hab)^
({{8{data_in[61][1]}} & 8'h7d)^
({{8{data_in[61][2]}} & 8'hfa)^
({{8{data_in[61][3]}} & 8'hdf)^
({{8{data_in[61][4]}} & 8'h95)^
({{8{data_in[61][5]}} & 8'h1)^
({{8{data_in[61][6]}} & 8'h2)^
({{8{data_in[61][7]}} & 8'h4)^
({{8{data_in[62][0]}} & 8'h53)^
({{8{data_in[62][1]}} & 8'ha6)^
({{8{data_in[62][2]}} & 8'h67)^
({{8{data_in[62][3]}} & 8'hce)^
({{8{data_in[62][4]}} & 8'hb7)^
({{8{data_in[62][5]}} & 8'h45)^
({{8{data_in[62][6]}} & 8'h8a)^
({{8{data_in[62][7]}} & 8'h3f)^
({{8{data_in[63][0]}} & 8'h94)^
({{8{data_in[63][1]}} & 8'h3)^
({{8{data_in[63][2]}} & 8'h6)^
({{8{data_in[63][3]}} & 8'hc)^
({{8{data_in[63][4]}} & 8'h18)^
({{8{data_in[63][5]}} & 8'h30)^
({{8{data_in[63][6]}} & 8'h60)^
({{8{data_in[63][7]}} & 8'hc0)^
({{8{data_in[64][0]}} & 8'h68)^
({{8{data_in[64][1]}} & 8'hd0)^
({{8{data_in[64][2]}} & 8'h8b)^
({{8{data_in[64][3]}} & 8'h3d)^
({{8{data_in[64][4]}} & 8'h7a)^
({{8{data_in[64][5]}} & 8'hf4)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[64][6]}} & 8'hc3)^
({{8{data_in[64][7]}} & 8'had)^
({{8{data_in[65][0]}} & 8'hb9)^
({{8{data_in[65][1]}} & 8'h59)^
({{8{data_in[65][2]}} & 8'hb2)^
({{8{data_in[65][3]}} & 8'h4f)^
({{8{data_in[65][4]}} & 8'h9e)^
({{8{data_in[65][5]}} & 8'h17)^
({{8{data_in[65][6]}} & 8'h2e)^
({{8{data_in[65][7]}} & 8'h5c)^
({{8{data_in[66][0]}} & 8'h64)^
({{8{data_in[66][1]}} & 8'hc8)^
({{8{data_in[66][2]}} & 8'hbb)^
({{8{data_in[66][3]}} & 8'h5d)^
({{8{data_in[66][4]}} & 8'hba)^
({{8{data_in[66][5]}} & 8'h5f)^
({{8{data_in[66][6]}} & 8'hbe)^
({{8{data_in[66][7]}} & 8'h57)^
({{8{data_in[67][0]}} & 8'h94)^
({{8{data_in[67][1]}} & 8'h3)^
({{8{data_in[67][2]}} & 8'h6)^
({{8{data_in[67][3]}} & 8'hc)^
({{8{data_in[67][4]}} & 8'h18)^
({{8{data_in[67][5]}} & 8'h30)^
({{8{data_in[67][6]}} & 8'h60)^
({{8{data_in[67][7]}} & 8'hc0)^
({{8{data_in[68][0]}} & 8'h92)^
({{8{data_in[68][1]}} & 8'hf)^
({{8{data_in[68][2]}} & 8'h1e)^
({{8{data_in[68][3]}} & 8'h3c)^
({{8{data_in[68][4]}} & 8'h78)^
({{8{data_in[68][5]}} & 8'hf0)^
({{8{data_in[68][6]}} & 8'hcb)^
({{8{data_in[68][7]}} & 8'hbd)^
({{8{data_in[69][0]}} & 8'h54)^
({{8{data_in[69][1]}} & 8'ha8)^
({{8{data_in[69][2]}} & 8'h7b)^
({{8{data_in[69][3]}} & 8'hf6)^
({{8{data_in[69][4]}} & 8'hc7)^
({{8{data_in[69][5]}} & 8'ha5)^
({{8{data_in[69][6]}} & 8'h61)^
({{8{data_in[69][7]}} & 8'hc2)^
({{8{data_in[70][0]}} & 8'h8d)^
({{8{data_in[70][1]}} & 8'h31)^
({{8{data_in[70][2]}} & 8'h62)^
({{8{data_in[70][3]}} & 8'hc4)^
({{8{data_in[70][4]}} & 8'ha3)^
({{8{data_in[70][5]}} & 8'h6d)^
({{8{data_in[70][6]}} & 8'hda)^
({{8{data_in[70][7]}} & 8'h9f)^
({{8{data_in[71][0]}} & 8'ha8)^
({{8{data_in[71][1]}} & 8'h7b)^
({{8{data_in[71][2]}} & 8'hf6)^
({{8{data_in[71][3]}} & 8'hc7)^
({{8{data_in[71][4]}} & 8'ha5)^
({{8{data_in[71][5]}} & 8'h61)^
({{8{data_in[71][6]}} & 8'hc2)^
({{8{data_in[71][7]}} & 8'haf)^
({{8{data_in[72][0]}} & 8'h84)^
({{8{data_in[72][1]}} & 8'h23)^
({{8{data_in[72][2]}} & 8'h46)^
({{8{data_in[72][3]}} & 8'h8c)^
({{8{data_in[72][4]}} & 8'h33)^
({{8{data_in[72][5]}} & 8'h66)^
({{8{data_in[72][6]}} & 8'hcc)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[72][7]}} & 8'hb3)^
({{8{data_in[73][0]}} & 8'h1f)^
({{8{data_in[73][1]}} & 8'h3e)^
({{8{data_in[73][2]}} & 8'h7c)^
({{8{data_in[73][3]}} & 8'hf8)^
({{8{data_in[73][4]}} & 8'hdb)^
({{8{data_in[73][5]}} & 8'h9d)^
({{8{data_in[73][6]}} & 8'h11)^
({{8{data_in[73][7]}} & 8'h22)^
({{8{data_in[74][0]}} & 8'h89)^
({{8{data_in[74][1]}} & 8'h39)^
({{8{data_in[74][2]}} & 8'h72)^
({{8{data_in[74][3]}} & 8'he4)^
({{8{data_in[74][4]}} & 8'he3)^
({{8{data_in[74][5]}} & 8'hed)^
({{8{data_in[74][6]}} & 8'hf1)^
({{8{data_in[74][7]}} & 8'hc9)^
({{8{data_in[75][0]}} & 8'h2e)^
({{8{data_in[75][1]}} & 8'h5c)^
({{8{data_in[75][2]}} & 8'hb8)^
({{8{data_in[75][3]}} & 8'h5b)^
({{8{data_in[75][4]}} & 8'hb6)^
({{8{data_in[75][5]}} & 8'h47)^
({{8{data_in[75][6]}} & 8'h8e)^
({{8{data_in[75][7]}} & 8'h37)^
({{8{data_in[76][0]}} & 8'hc8)^
({{8{data_in[76][1]}} & 8'hbb)^
({{8{data_in[76][2]}} & 8'h5d)^
({{8{data_in[76][3]}} & 8'hba)^
({{8{data_in[76][4]}} & 8'h5f)^
({{8{data_in[76][5]}} & 8'hbe)^
({{8{data_in[76][6]}} & 8'h57)^
({{8{data_in[76][7]}} & 8'hae)^
({{8{data_in[77][0]}} & 8'hdd)^
({{8{data_in[77][1]}} & 8'h91)^
({{8{data_in[77][2]}} & 8'h9)^
({{8{data_in[77][3]}} & 8'h12)^
({{8{data_in[77][4]}} & 8'h24)^
({{8{data_in[77][5]}} & 8'h48)^
({{8{data_in[77][6]}} & 8'h90)^
({{8{data_in[77][7]}} & 8'hb)^
({{8{data_in[78][0]}} & 8'hd9)^
({{8{data_in[78][1]}} & 8'h99)^
({{8{data_in[78][2]}} & 8'h19)^
({{8{data_in[78][3]}} & 8'h32)^
({{8{data_in[78][4]}} & 8'h64)^
({{8{data_in[78][5]}} & 8'hc8)^
({{8{data_in[78][6]}} & 8'hbb)^
({{8{data_in[78][7]}} & 8'h5d)^
({{8{data_in[79][0]}} & 8'hd0)^
({{8{data_in[79][1]}} & 8'h8b)^
({{8{data_in[79][2]}} & 8'h3d)^
({{8{data_in[79][3]}} & 8'h7a)^
({{8{data_in[79][4]}} & 8'hf4)^
({{8{data_in[79][5]}} & 8'hc3)^
({{8{data_in[79][6]}} & 8'had)^
({{8{data_in[79][7]}} & 8'h71)^
({{8{data_in[80][0]}} & 8'hd4)^
({{8{data_in[80][1]}} & 8'h83)^
({{8{data_in[80][2]}} & 8'h2d)^
({{8{data_in[80][3]}} & 8'h5a)^
({{8{data_in[80][4]}} & 8'hb4)^
({{8{data_in[80][5]}} & 8'h43)^
({{8{data_in[80][6]}} & 8'h86)^
({{8{data_in[80][7]}} & 8'h27})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[81][0]}} & 8'h19)^
({{8{data_in[81][1]}} & 8'h32)^
({{8{data_in[81][2]}} & 8'h64)^
({{8{data_in[81][3]}} & 8'hc8)^
({{8{data_in[81][4]}} & 8'hbb)^
({{8{data_in[81][5]}} & 8'h5d)^
({{8{data_in[81][6]}} & 8'hba)^
({{8{data_in[81][7]}} & 8'h5f)^
({{8{data_in[82][0]}} & 8'hf6)^
({{8{data_in[82][1]}} & 8'hc7)^
({{8{data_in[82][2]}} & 8'ha5)^
({{8{data_in[82][3]}} & 8'h61)^
({{8{data_in[82][4]}} & 8'hc2)^
({{8{data_in[82][5]}} & 8'haf)^
({{8{data_in[82][6]}} & 8'h75)^
({{8{data_in[82][7]}} & 8'hea)^
({{8{data_in[83][0]}} & 8'hc0)^
({{8{data_in[83][1]}} & 8'hab)^
({{8{data_in[83][2]}} & 8'h7d)^
({{8{data_in[83][3]}} & 8'hfa)^
({{8{data_in[83][4]}} & 8'hdf)^
({{8{data_in[83][5]}} & 8'h95)^
({{8{data_in[83][6]}} & 8'h1)^
({{8{data_in[83][7]}} & 8'h2)^
({{8{data_in[84][0]}} & 8'hc4)^
({{8{data_in[84][1]}} & 8'ha3)^
({{8{data_in[84][2]}} & 8'h6d)^
({{8{data_in[84][3]}} & 8'hda)^
({{8{data_in[84][4]}} & 8'h9f)^
({{8{data_in[84][5]}} & 8'h15)^
({{8{data_in[84][6]}} & 8'h2a)^
({{8{data_in[84][7]}} & 8'h54)^
({{8{data_in[85][0]}} & 8'hb6)^
({{8{data_in[85][1]}} & 8'h47)^
({{8{data_in[85][2]}} & 8'h8e)^
({{8{data_in[85][3]}} & 8'h37)^
({{8{data_in[85][4]}} & 8'h6e)^
({{8{data_in[85][5]}} & 8'hdc)^
({{8{data_in[85][6]}} & 8'h93)^
({{8{data_in[85][7]}} & 8'hd)^
({{8{data_in[86][0]}} & 8'hac)^
({{8{data_in[86][1]}} & 8'h73)^
({{8{data_in[86][2]}} & 8'he6)^
({{8{data_in[86][3]}} & 8'he7)^
({{8{data_in[86][4]}} & 8'he5)^
({{8{data_in[86][5]}} & 8'he1)^
({{8{data_in[86][6]}} & 8'he9)^
({{8{data_in[86][7]}} & 8'hf9)^
({{8{data_in[87][0]}} & 8'h55)^
({{8{data_in[87][1]}} & 8'haa)^
({{8{data_in[87][2]}} & 8'h7f)^
({{8{data_in[87][3]}} & 8'hfe)^
({{8{data_in[87][4]}} & 8'hd7)^
({{8{data_in[87][5]}} & 8'h85)^
({{8{data_in[87][6]}} & 8'h21)^
({{8{data_in[87][7]}} & 8'h42)^
({{8{data_in[88][0]}} & 8'hbf)^
({{8{data_in[88][1]}} & 8'h55)^
({{8{data_in[88][2]}} & 8'haa)^
({{8{data_in[88][3]}} & 8'h7f)^
({{8{data_in[88][4]}} & 8'hfe)^
({{8{data_in[88][5]}} & 8'hd7)^
({{8{data_in[88][6]}} & 8'h85)^
({{8{data_in[88][7]}} & 8'h21)^
({{8{data_in[89][0]}} & 8'hff)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[89][1]}} & 8'hd5)^
({{8{data_in[89][2]}} & 8'h81)^
({{8{data_in[89][3]}} & 8'h29)^
({{8{data_in[89][4]}} & 8'h52)^
({{8{data_in[89][5]}} & 8'ha4)^
({{8{data_in[89][6]}} & 8'h63)^
({{8{data_in[89][7]}} & 8'hc6)^
({{8{data_in[90][0]}} & 8'h2e)^
({{8{data_in[90][1]}} & 8'h5c)^
({{8{data_in[90][2]}} & 8'hb8)^
({{8{data_in[90][3]}} & 8'h5b)^
({{8{data_in[90][4]}} & 8'hb6)^
({{8{data_in[90][5]}} & 8'h47)^
({{8{data_in[90][6]}} & 8'h8e)^
({{8{data_in[90][7]}} & 8'h37)^
({{8{data_in[91][0]}} & 8'hcc)^
({{8{data_in[91][1]}} & 8'hb3)^
({{8{data_in[91][2]}} & 8'h4d)^
({{8{data_in[91][3]}} & 8'h9a)^
({{8{data_in[91][4]}} & 8'h1f)^
({{8{data_in[91][5]}} & 8'h3e)^
({{8{data_in[91][6]}} & 8'h7c)^
({{8{data_in[91][7]}} & 8'hf8)^
({{8{data_in[92][0]}} & 8'hd1)^
({{8{data_in[92][1]}} & 8'h89)^
({{8{data_in[92][2]}} & 8'h39)^
({{8{data_in[92][3]}} & 8'h72)^
({{8{data_in[92][4]}} & 8'he4)^
({{8{data_in[92][5]}} & 8'he3)^
({{8{data_in[92][6]}} & 8'hd})^
({{8{data_in[92][7]}} & 8'hf1)^
({{8{data_in[93][0]}} & 8'hb0)^
({{8{data_in[93][1]}} & 8'h4b)^
({{8{data_in[93][2]}} & 8'h96)^
({{8{data_in[93][3]}} & 8'h7)^
({{8{data_in[93][4]}} & 8'he)^
({{8{data_in[93][5]}} & 8'h1c)^
({{8{data_in[93][6]}} & 8'h38)^
({{8{data_in[93][7]}} & 8'h70)^
({{8{data_in[94][0]}} & 8'hd)^
({{8{data_in[94][1]}} & 8'h1a)^
({{8{data_in[94][2]}} & 8'h34)^
({{8{data_in[94][3]}} & 8'h68)^
({{8{data_in[94][4]}} & 8'hd0)^
({{8{data_in[94][5]}} & 8'h8b)^
({{8{data_in[94][6]}} & 8'h3d)^
({{8{data_in[94][7]}} & 8'h7a)^
({{8{data_in[95][0]}} & 8'h84)^
({{8{data_in[95][1]}} & 8'h23)^
({{8{data_in[95][2]}} & 8'h46)^
({{8{data_in[95][3]}} & 8'h8c)^
({{8{data_in[95][4]}} & 8'h33)^
({{8{data_in[95][5]}} & 8'h66)^
({{8{data_in[95][6]}} & 8'hcc)^
({{8{data_in[95][7]}} & 8'hb3)^
({{8{data_in[96][0]}} & 8'h91)^
({{8{data_in[96][1]}} & 8'h9)^
({{8{data_in[96][2]}} & 8'h12)^
({{8{data_in[96][3]}} & 8'h24)^
({{8{data_in[96][4]}} & 8'h48)^
({{8{data_in[96][5]}} & 8'h90)^
({{8{data_in[96][6]}} & 8'hb)^
({{8{data_in[96][7]}} & 8'h16)^
({{8{data_in[97][0]}} & 8'hcf)^
({{8{data_in[97][1]}} & 8'hb5})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[97][2]}} & 8'h41)^
({{8{data_in[97][3]}} & 8'h82)^
({{8{data_in[97][4]}} & 8'h2f)^
({{8{data_in[97][5]}} & 8'h5e)^
({{8{data_in[97][6]}} & 8'hbc)^
({{8{data_in[97][7]}} & 8'h53)^
({{8{data_in[98][0]}} & 8'h30)^
({{8{data_in[98][1]}} & 8'h60)^
({{8{data_in[98][2]}} & 8'hc0)^
({{8{data_in[98][3]}} & 8'hab)^
({{8{data_in[98][4]}} & 8'h7d)^
({{8{data_in[98][5]}} & 8'hfa)^
({{8{data_in[98][6]}} & 8'hdf)^
({{8{data_in[98][7]}} & 8'h95)^
({{8{data_in[99][0]}} & 8'he5)^
({{8{data_in[99][1]}} & 8'he1)^
({{8{data_in[99][2]}} & 8'he9)^
({{8{data_in[99][3]}} & 8'hf9)^
({{8{data_in[99][4]}} & 8'hd9)^
({{8{data_in[99][5]}} & 8'h99)^
({{8{data_in[99][6]}} & 8'h19)^
({{8{data_in[99][7]}} & 8'h32)^
({{8{data_in[100][0]}} & 8'h98)^
({{8{data_in[100][1]}} & 8'h1b)^
({{8{data_in[100][2]}} & 8'h36)^
({{8{data_in[100][3]}} & 8'h6c)^
({{8{data_in[100][4]}} & 8'hd8)^
({{8{data_in[100][5]}} & 8'h9b)^
({{8{data_in[100][6]}} & 8'h1d)^
({{8{data_in[100][7]}} & 8'h3a)^
({{8{data_in[101][0]}} & 8'hfe)^
({{8{data_in[101][1]}} & 8'hd7)^
({{8{data_in[101][2]}} & 8'h85)^
({{8{data_in[101][3]}} & 8'h21)^
({{8{data_in[101][4]}} & 8'h42)^
({{8{data_in[101][5]}} & 8'h84)^
({{8{data_in[101][6]}} & 8'h23)^
({{8{data_in[101][7]}} & 8'h46)^
({{8{data_in[102][0]}} & 8'hf3)^
({{8{data_in[102][1]}} & 8'hcd)^
({{8{data_in[102][2]}} & 8'hb1)^
({{8{data_in[102][3]}} & 8'h49)^
({{8{data_in[102][4]}} & 8'h92)^
({{8{data_in[102][5]}} & 8'hf)^
({{8{data_in[102][6]}} & 8'h1e)^
({{8{data_in[102][7]}} & 8'h3c)^
({{8{data_in[103][0]}} & 8'h1e)^
({{8{data_in[103][1]}} & 8'h3c)^
({{8{data_in[103][2]}} & 8'h78)^
({{8{data_in[103][3]}} & 8'hf0)^
({{8{data_in[103][4]}} & 8'hcb)^
({{8{data_in[103][5]}} & 8'hbd)^
({{8{data_in[103][6]}} & 8'h51)^
({{8{data_in[103][7]}} & 8'ha2)^
({{8{data_in[104][0]}} & 8'h18)^
({{8{data_in[104][1]}} & 8'h30)^
({{8{data_in[104][2]}} & 8'h60)^
({{8{data_in[104][3]}} & 8'hc0)^
({{8{data_in[104][4]}} & 8'hab)^
({{8{data_in[104][5]}} & 8'h7d)^
({{8{data_in[104][6]}} & 8'hfa)^
({{8{data_in[104][7]}} & 8'hdf)^
({{8{data_in[105][0]}} & 8'hb8)^
({{8{data_in[105][1]}} & 8'h5b)^
({{8{data_in[105][2]}} & 8'hb6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[105][3]}} & 8'h47)^
({{8{data_in[105][4]}} & 8'h8e)^
({{8{data_in[105][5]}} & 8'h37)^
({{8{data_in[105][6]}} & 8'h6e)^
({{8{data_in[105][7]}} & 8'hdcc)^
({{8{data_in[106][0]}} & 8'h8c)^
({{8{data_in[106][1]}} & 8'h33)^
({{8{data_in[106][2]}} & 8'h66)^
({{8{data_in[106][3]}} & 8'hcc)^
({{8{data_in[106][4]}} & 8'hb3)^
({{8{data_in[106][5]}} & 8'h4d)^
({{8{data_in[106][6]}} & 8'h9a)^
({{8{data_in[106][7]}} & 8'h1f)^
({{8{data_in[107][0]}} & 8'h98)^
({{8{data_in[107][1]}} & 8'h1b)^
({{8{data_in[107][2]}} & 8'h36)^
({{8{data_in[107][3]}} & 8'h6c)^
({{8{data_in[107][4]}} & 8'hd8)^
({{8{data_in[107][5]}} & 8'h9b)^
({{8{data_in[107][6]}} & 8'h1d)^
({{8{data_in[107][7]}} & 8'h3a)^
({{8{data_in[108][0]}} & 8'h94)^
({{8{data_in[108][1]}} & 8'h3)^
({{8{data_in[108][2]}} & 8'h6)^
({{8{data_in[108][3]}} & 8'hc)^
({{8{data_in[108][4]}} & 8'h18)^
({{8{data_in[108][5]}} & 8'h30)^
({{8{data_in[108][6]}} & 8'h60)^
({{8{data_in[108][7]}} & 8'hc0)^
({{8{data_in[109][0]}} & 8'hef)^
({{8{data_in[109][1]}} & 8'hf5)^
({{8{data_in[109][2]}} & 8'hc1)^
({{8{data_in[109][3]}} & 8'ha9)^
({{8{data_in[109][4]}} & 8'h79)^
({{8{data_in[109][5]}} & 8'hf2)^
({{8{data_in[109][6]}} & 8'hcfc)^
({{8{data_in[109][7]}} & 8'hb5)^
({{8{data_in[110][0]}} & 8'h4e)^
({{8{data_in[110][1]}} & 8'h9c)^
({{8{data_in[110][2]}} & 8'h13)^
({{8{data_in[110][3]}} & 8'h26)^
({{8{data_in[110][4]}} & 8'h4c)^
({{8{data_in[110][5]}} & 8'h98)^
({{8{data_in[110][6]}} & 8'h1b)^
({{8{data_in[110][7]}} & 8'h36)^
({{8{data_in[111][0]}} & 8'hc)^
({{8{data_in[111][1]}} & 8'h18)^
({{8{data_in[111][2]}} & 8'h30)^
({{8{data_in[111][3]}} & 8'h60)^
({{8{data_in[111][4]}} & 8'hc0)^
({{8{data_in[111][5]}} & 8'hab)^
({{8{data_in[111][6]}} & 8'h7d)^
({{8{data_in[111][7]}} & 8'hfa)^
({{8{data_in[112][0]}} & 8'h75)^
({{8{data_in[112][1]}} & 8'hea)^
({{8{data_in[112][2]}} & 8'hff)^
({{8{data_in[112][3]}} & 8'hd5)^
({{8{data_in[112][4]}} & 8'h81)^
({{8{data_in[112][5]}} & 8'h29)^
({{8{data_in[112][6]}} & 8'h52)^
({{8{data_in[112][7]}} & 8'ha4)^
({{8{data_in[113][0]}} & 8'h35)^
({{8{data_in[113][1]}} & 8'h6a)^
({{8{data_in[113][2]}} & 8'hd4)^
({{8{data_in[113][3]}} & 8'h83)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[113][4]}} & 8'h2d)^
({{8{data_in[113][5]}} & 8'h5a)^
({{8{data_in[113][6]}} & 8'hb4)^
({{8{data_in[113][7]}} & 8'h43)^
({{8{data_in[114][0]}} & 8'h18)^
({{8{data_in[114][1]}} & 8'h30)^
({{8{data_in[114][2]}} & 8'h60)^
({{8{data_in[114][3]}} & 8'hc0)^
({{8{data_in[114][4]}} & 8'hab)^
({{8{data_in[114][5]}} & 8'h7d)^
({{8{data_in[114][6]}} & 8'hfa)^
({{8{data_in[114][7]}} & 8'hdf)^
({{8{data_in[115][0]}} & 8'ha0)^
({{8{data_in[115][1]}} & 8'h6b)^
({{8{data_in[115][2]}} & 8'hd6)^
({{8{data_in[115][3]}} & 8'h87)^
({{8{data_in[115][4]}} & 8'h25)^
({{8{data_in[115][5]}} & 8'h4a)^
({{8{data_in[115][6]}} & 8'h94)^
({{8{data_in[115][7]}} & 8'h3)^
({{8{data_in[116][0]}} & 8'h41)^
({{8{data_in[116][1]}} & 8'h82)^
({{8{data_in[116][2]}} & 8'h2f)^
({{8{data_in[116][3]}} & 8'h5e)^
({{8{data_in[116][4]}} & 8'hbc)^
({{8{data_in[116][5]}} & 8'h53)^
({{8{data_in[116][6]}} & 8'ha6)^
({{8{data_in[116][7]}} & 8'h67)^
({{8{data_in[117][0]}} & 8'he5)^
({{8{data_in[117][1]}} & 8'he1)^
({{8{data_in[117][2]}} & 8'he9)^
({{8{data_in[117][3]}} & 8'hf9)^
({{8{data_in[117][4]}} & 8'hd9)^
({{8{data_in[117][5]}} & 8'h99)^
({{8{data_in[117][6]}} & 8'h19)^
({{8{data_in[117][7]}} & 8'h32)^
({{8{data_in[118][0]}} & 8'hd3)^
({{8{data_in[118][1]}} & 8'h8d)^
({{8{data_in[118][2]}} & 8'h31)^
({{8{data_in[118][3]}} & 8'h62)^
({{8{data_in[118][4]}} & 8'hc4)^
({{8{data_in[118][5]}} & 8'ha3)^
({{8{data_in[118][6]}} & 8'h6d)^
({{8{data_in[118][7]}} & 8'hda)^
({{8{data_in[119][0]}} & 8'h22)^
({{8{data_in[119][1]}} & 8'h44)^
({{8{data_in[119][2]}} & 8'h88)^
({{8{data_in[119][3]}} & 8'h3b)^
({{8{data_in[119][4]}} & 8'h76)^
({{8{data_in[119][5]}} & 8'hec)^
({{8{data_in[119][6]}} & 8'hf3)^
({{8{data_in[119][7]}} & 8'hcd)^
({{8{data_in[120][0]}} & 8'hbd)^
({{8{data_in[120][1]}} & 8'h51)^
({{8{data_in[120][2]}} & 8'ha2)^
({{8{data_in[120][3]}} & 8'h6f)^
({{8{data_in[120][4]}} & 8'hde)^
({{8{data_in[120][5]}} & 8'h97)^
({{8{data_in[120][6]}} & 8'h5)^
({{8{data_in[120][7]}} & 8'ha)^
({{8{data_in[121][0]}} & 8'hf7)^
({{8{data_in[121][1]}} & 8'hc5)^
({{8{data_in[121][2]}} & 8'ha1)^
({{8{data_in[121][3]}} & 8'h69)^
({{8{data_in[121][4]}} & 8'hd2)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[121][5]}} & 8'h8f)^
({{8{data_in[121][6]}} & 8'h35)^
({{8{data_in[121][7]}} & 8'h6a)^
({{8{data_in[122][0]}} & 8'h25)^
({{8{data_in[122][1]}} & 8'h4a)^
({{8{data_in[122][2]}} & 8'h94)^
({{8{data_in[122][3]}} & 8'h3)^
({{8{data_in[122][4]}} & 8'h6)^
({{8{data_in[122][5]}} & 8'hc)^
({{8{data_in[122][6]}} & 8'h18)^
({{8{data_in[122][7]}} & 8'h30)^
({{8{data_in[123][0]}} & 8'h4e)^
({{8{data_in[123][1]}} & 8'h9c)^
({{8{data_in[123][2]}} & 8'h13)^
({{8{data_in[123][3]}} & 8'h26)^
({{8{data_in[123][4]}} & 8'h4c)^
({{8{data_in[123][5]}} & 8'h98)^
({{8{data_in[123][6]}} & 8'h1b)^
({{8{data_in[123][7]}} & 8'h36)^
({{8{data_in[124][0]}} & 8'h2e)^
({{8{data_in[124][1]}} & 8'h5c)^
({{8{data_in[124][2]}} & 8'hb8)^
({{8{data_in[124][3]}} & 8'h5b)^
({{8{data_in[124][4]}} & 8'hb6)^
({{8{data_in[124][5]}} & 8'h47)^
({{8{data_in[124][6]}} & 8'h8e)^
({{8{data_in[124][7]}} & 8'h37)^
({{8{data_in[125][0]}} & 8'hd5)^
({{8{data_in[125][1]}} & 8'h81)^
({{8{data_in[125][2]}} & 8'h29)^
({{8{data_in[125][3]}} & 8'h52)^
({{8{data_in[125][4]}} & 8'ha4)^
({{8{data_in[125][5]}} & 8'h63)^
({{8{data_in[125][6]}} & 8'hc6)^
({{8{data_in[125][7]}} & 8'ha7)^
({{8{data_in[126][0]}} & 8'hbf)^
({{8{data_in[126][1]}} & 8'h55)^
({{8{data_in[126][2]}} & 8'haa)^
({{8{data_in[126][3]}} & 8'h7f)^
({{8{data_in[126][4]}} & 8'hfe)^
({{8{data_in[126][5]}} & 8'hd7)^
({{8{data_in[126][6]}} & 8'h85)^
({{8{data_in[126][7]}} & 8'h21)^
({{8{data_in[127][0]}} & 8'h40)^
({{8{data_in[127][1]}} & 8'h80)^
({{8{data_in[127][2]}} & 8'h2b)^
({{8{data_in[127][3]}} & 8'h56)^
({{8{data_in[127][4]}} & 8'hac)^
({{8{data_in[127][5]}} & 8'h73)^
({{8{data_in[127][6]}} & 8'he6)^
({{8{data_in[127][7]}} & 8'he7)^
({{8{data_in[128][0]}} & 8'h8a)^
({{8{data_in[128][1]}} & 8'h3f)^
({{8{data_in[128][2]}} & 8'h7e)^
({{8{data_in[128][3]}} & 8'hfc)^
({{8{data_in[128][4]}} & 8'hd3)^
({{8{data_in[128][5]}} & 8'h8d)^
({{8{data_in[128][6]}} & 8'h31)^
({{8{data_in[128][7]}} & 8'h62)^
({{8{data_in[129][0]}} & 8'h18)^
({{8{data_in[129][1]}} & 8'h30)^
({{8{data_in[129][2]}} & 8'h60)^
({{8{data_in[129][3]}} & 8'hc0)^
({{8{data_in[129][4]}} & 8'hab)^
({{8{data_in[129][5]}} & 8'h7d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[129][6]}} & 8'hfa)^
({{8{data_in[129][7]}} & 8'hdf)^
({{8{data_in[130][0]}} & 8'hdc)^
({{8{data_in[130][1]}} & 8'h93)^
({{8{data_in[130][2]}} & 8'hd)^
({{8{data_in[130][3]}} & 8'h1a)^
({{8{data_in[130][4]}} & 8'h34)^
({{8{data_in[130][5]}} & 8'h68)^
({{8{data_in[130][6]}} & 8'hd0)^
({{8{data_in[130][7]}} & 8'h8b)^
({{8{data_in[131][0]}} & 8'ha0)^
({{8{data_in[131][1]}} & 8'h6b)^
({{8{data_in[131][2]}} & 8'hd6)^
({{8{data_in[131][3]}} & 8'h87)^
({{8{data_in[131][4]}} & 8'h25)^
({{8{data_in[131][5]}} & 8'h4a)^
({{8{data_in[131][6]}} & 8'h94)^
({{8{data_in[131][7]}} & 8'h3)^
({{8{data_in[132][0]}} & 8'hb1)^
({{8{data_in[132][1]}} & 8'h49)^
({{8{data_in[132][2]}} & 8'h92)^
({{8{data_in[132][3]}} & 8'hf)^
({{8{data_in[132][4]}} & 8'h1e)^
({{8{data_in[132][5]}} & 8'h3c)^
({{8{data_in[132][6]}} & 8'h78)^
({{8{data_in[132][7]}} & 8'hf0)^
({{8{data_in[133][0]}} & 8'h32)^
({{8{data_in[133][1]}} & 8'h64)^
({{8{data_in[133][2]}} & 8'hc8)^
({{8{data_in[133][3]}} & 8'hbb)^
({{8{data_in[133][4]}} & 8'h5d)^
({{8{data_in[133][5]}} & 8'hba)^
({{8{data_in[133][6]}} & 8'h5f)^
({{8{data_in[133][7]}} & 8'hbe)^
({{8{data_in[134][0]}} & 8'h8c)^
({{8{data_in[134][1]}} & 8'h33)^
({{8{data_in[134][2]}} & 8'h66)^
({{8{data_in[134][3]}} & 8'hcc)^
({{8{data_in[134][4]}} & 8'hb3)^
({{8{data_in[134][5]}} & 8'h4d)^
({{8{data_in[134][6]}} & 8'h9a)^
({{8{data_in[134][7]}} & 8'h1f)^
({{8{data_in[135][0]}} & 8'h6e)^
({{8{data_in[135][1]}} & 8'hdc)^
({{8{data_in[135][2]}} & 8'h93)^
({{8{data_in[135][3]}} & 8'hd)^
({{8{data_in[135][4]}} & 8'h1a)^
({{8{data_in[135][5]}} & 8'h34)^
({{8{data_in[135][6]}} & 8'h68)^
({{8{data_in[135][7]}} & 8'hd0)^
({{8{data_in[136][0]}} & 8'h83)^
({{8{data_in[136][1]}} & 8'h2d)^
({{8{data_in[136][2]}} & 8'h5a)^
({{8{data_in[136][3]}} & 8'hb4)^
({{8{data_in[136][4]}} & 8'h43)^
({{8{data_in[136][5]}} & 8'h86)^
({{8{data_in[136][6]}} & 8'h27)^
({{8{data_in[136][7]}} & 8'h4e)^
({{8{data_in[137][0]}} & 8'h8a)^
({{8{data_in[137][1]}} & 8'h3f)^
({{8{data_in[137][2]}} & 8'h7e)^
({{8{data_in[137][3]}} & 8'hfc)^
({{8{data_in[137][4]}} & 8'hd3)^
({{8{data_in[137][5]}} & 8'h8d)^
({{8{data_in[137][6]}} & 8'h31)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[137][7]}} & 8'h62)^
({{8{data_in[138][0]}} & 8'hf4)^
({{8{data_in[138][1]}} & 8'hc3)^
({{8{data_in[138][2]}} & 8'had)^
({{8{data_in[138][3]}} & 8'h71)^
({{8{data_in[138][4]}} & 8'he2)^
({{8{data_in[138][5]}} & 8'hef)^
({{8{data_in[138][6]}} & 8'hf5)^
({{8{data_in[138][7]}} & 8'hc1)^
({{8{data_in[139][0]}} & 8'hbe)^
({{8{data_in[139][1]}} & 8'h57)^
({{8{data_in[139][2]}} & 8'hae)^
({{8{data_in[139][3]}} & 8'h77)^
({{8{data_in[139][4]}} & 8'hee)^
({{8{data_in[139][5]}} & 8'hf7)^
({{8{data_in[139][6]}} & 8'hc5)^
({{8{data_in[139][7]}} & 8'ha1)^
({{8{data_in[140][0]}} & 8'h40)^
({{8{data_in[140][1]}} & 8'h80)^
({{8{data_in[140][2]}} & 8'h2b)^
({{8{data_in[140][3]}} & 8'h56)^
({{8{data_in[140][4]}} & 8'hac)^
({{8{data_in[140][5]}} & 8'h73)^
({{8{data_in[140][6]}} & 8'he6)^
({{8{data_in[140][7]}} & 8'he7)^
({{8{data_in[141][0]}} & 8'h61)^
({{8{data_in[141][1]}} & 8'hc2)^
({{8{data_in[141][2]}} & 8'haf)^
({{8{data_in[141][3]}} & 8'h75)^
({{8{data_in[141][4]}} & 8'hea)^
({{8{data_in[141][5]}} & 8'hff)^
({{8{data_in[141][6]}} & 8'hd5)^
({{8{data_in[141][7]}} & 8'h81)^
({{8{data_in[142][0]}} & 8'he3)^
({{8{data_in[142][1]}} & 8'hed)^
({{8{data_in[142][2]}} & 8'hf1)^
({{8{data_in[142][3]}} & 8'hc9)^
({{8{data_in[142][4]}} & 8'hb9)^
({{8{data_in[142][5]}} & 8'h59)^
({{8{data_in[142][6]}} & 8'hb2)^
({{8{data_in[142][7]}} & 8'h4f)^
({{8{data_in[143][0]}} & 8'h8b)^
({{8{data_in[143][1]}} & 8'h3d)^
({{8{data_in[143][2]}} & 8'h7a)^
({{8{data_in[143][3]}} & 8'hf4)^
({{8{data_in[143][4]}} & 8'hc3)^
({{8{data_in[143][5]}} & 8'had)^
({{8{data_in[143][6]}} & 8'h71)^
({{8{data_in[143][7]}} & 8'he2)^
({{8{data_in[144][0]}} & 8'h5d)^
({{8{data_in[144][1]}} & 8'hba)^
({{8{data_in[144][2]}} & 8'h5f)^
({{8{data_in[144][3]}} & 8'hbe)^
({{8{data_in[144][4]}} & 8'h57)^
({{8{data_in[144][5]}} & 8'hae)^
({{8{data_in[144][6]}} & 8'h77)^
({{8{data_in[144][7]}} & 8'hee)^
({{8{data_in[145][0]}} & 8'ha8)^
({{8{data_in[145][1]}} & 8'h7b)^
({{8{data_in[145][2]}} & 8'hf6)^
({{8{data_in[145][3]}} & 8'hc7)^
({{8{data_in[145][4]}} & 8'ha5)^
({{8{data_in[145][5]}} & 8'h61)^
({{8{data_in[145][6]}} & 8'hc2)^
({{8{data_in[145][7]}} & 8'haf)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[146][0]}} & 8'h69)^
({{8{data_in[146][1]}} & 8'hd2)^
({{8{data_in[146][2]}} & 8'h8f)^
({{8{data_in[146][3]}} & 8'h35)^
({{8{data_in[146][4]}} & 8'h6a)^
({{8{data_in[146][5]}} & 8'hd4)^
({{8{data_in[146][6]}} & 8'h83)^
({{8{data_in[146][7]}} & 8'h2d)^
({{8{data_in[147][0]}} & 8'h10)^
({{8{data_in[147][1]}} & 8'h20)^
({{8{data_in[147][2]}} & 8'h40)^
({{8{data_in[147][3]}} & 8'h80)^
({{8{data_in[147][4]}} & 8'h2b)^
({{8{data_in[147][5]}} & 8'h56)^
({{8{data_in[147][6]}} & 8'hac)^
({{8{data_in[147][7]}} & 8'h73)^
({{8{data_in[148][0]}} & 8'hff)^
({{8{data_in[148][1]}} & 8'hd5)^
({{8{data_in[148][2]}} & 8'h81)^
({{8{data_in[148][3]}} & 8'h29)^
({{8{data_in[148][4]}} & 8'h52)^
({{8{data_in[148][5]}} & 8'ha4)^
({{8{data_in[148][6]}} & 8'h63)^
({{8{data_in[148][7]}} & 8'hc6)^
({{8{data_in[149][0]}} & 8'h6e)^
({{8{data_in[149][1]}} & 8'hdc)^
({{8{data_in[149][2]}} & 8'h93)^
({{8{data_in[149][3]}} & 8'hd)^
({{8{data_in[149][4]}} & 8'h1a)^
({{8{data_in[149][5]}} & 8'h34)^
({{8{data_in[149][6]}} & 8'h68)^
({{8{data_in[149][7]}} & 8'hd0)^
({{8{data_in[150][0]}} & 8'hd1)^
({{8{data_in[150][1]}} & 8'h89)^
({{8{data_in[150][2]}} & 8'h39)^
({{8{data_in[150][3]}} & 8'h72)^
({{8{data_in[150][4]}} & 8'he4)^
({{8{data_in[150][5]}} & 8'he3)^
({{8{data_in[150][6]}} & 8'hed)^
({{8{data_in[150][7]}} & 8'hf1)^
({{8{data_in[151][0]}} & 8'h53)^
({{8{data_in[151][1]}} & 8'ha6)^
({{8{data_in[151][2]}} & 8'h67)^
({{8{data_in[151][3]}} & 8'hce)^
({{8{data_in[151][4]}} & 8'hb7)^
({{8{data_in[151][5]}} & 8'h45)^
({{8{data_in[151][6]}} & 8'h8a)^
({{8{data_in[151][7]}} & 8'h3f)^
({{8{data_in[152][0]}} & 8'h74)^
({{8{data_in[152][1]}} & 8'he8)^
({{8{data_in[152][2]}} & 8'hfb)^
({{8{data_in[152][3]}} & 8'hdd)^
({{8{data_in[152][4]}} & 8'h91)^
({{8{data_in[152][5]}} & 8'h9)^
({{8{data_in[152][6]}} & 8'h12)^
({{8{data_in[152][7]}} & 8'h24)^
({{8{data_in[153][0]}} & 8'h90)^
({{8{data_in[153][1]}} & 8'hb)^
({{8{data_in[153][2]}} & 8'h16)^
({{8{data_in[153][3]}} & 8'h2c)^
({{8{data_in[153][4]}} & 8'h58)^
({{8{data_in[153][5]}} & 8'hb0)^
({{8{data_in[153][6]}} & 8'h4b)^
({{8{data_in[153][7]}} & 8'h96)^
({{8{data_in[154][0]}} & 8'h5b)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[154][1]}} & 8'hb6)^
({{8{data_in[154][2]}} & 8'h47)^
({{8{data_in[154][3]}} & 8'h8e)^
({{8{data_in[154][4]}} & 8'h37)^
({{8{data_in[154][5]}} & 8'h6e)^
({{8{data_in[154][6]}} & 8'hdc)^
({{8{data_in[154][7]}} & 8'h93)^
({{8{data_in[155][0]}} & 8'h48)^
({{8{data_in[155][1]}} & 8'h90)^
({{8{data_in[155][2]}} & 8'hb)^
({{8{data_in[155][3]}} & 8'h16)^
({{8{data_in[155][4]}} & 8'h2c)^
({{8{data_in[155][5]}} & 8'h58)^
({{8{data_in[155][6]}} & 8'hb0)^
({{8{data_in[155][7]}} & 8'h4b)^
({{8{data_in[156][0]}} & 8'hd1)^
({{8{data_in[156][1]}} & 8'h89)^
({{8{data_in[156][2]}} & 8'h39)^
({{8{data_in[156][3]}} & 8'h72)^
({{8{data_in[156][4]}} & 8'he4)^
({{8{data_in[156][5]}} & 8'he3)^
({{8{data_in[156][6]}} & 8'hed)^
({{8{data_in[156][7]}} & 8'hf1)^
({{8{data_in[157][0]}} & 8'he6)^
({{8{data_in[157][1]}} & 8'he7)^
({{8{data_in[157][2]}} & 8'he5)^
({{8{data_in[157][3]}} & 8'he1)^
({{8{data_in[157][4]}} & 8'he9)^
({{8{data_in[157][5]}} & 8'hf9)^
({{8{data_in[157][6]}} & 8'hd9)^
({{8{data_in[157][7]}} & 8'h99)^
({{8{data_in[158][0]}} & 8'h41)^
({{8{data_in[158][1]}} & 8'h82)^
({{8{data_in[158][2]}} & 8'h2f)^
({{8{data_in[158][3]}} & 8'h5e)^
({{8{data_in[158][4]}} & 8'hbc)^
({{8{data_in[158][5]}} & 8'h53)^
({{8{data_in[158][6]}} & 8'ha6)^
({{8{data_in[158][7]}} & 8'h67)^
({{8{data_in[159][0]}} & 8'hb7)^
({{8{data_in[159][1]}} & 8'h45)^
({{8{data_in[159][2]}} & 8'h8a)^
({{8{data_in[159][3]}} & 8'h3f)^
({{8{data_in[159][4]}} & 8'h7e)^
({{8{data_in[159][5]}} & 8'hfc)^
({{8{data_in[159][6]}} & 8'hd3)^
({{8{data_in[159][7]}} & 8'h8d)^
({{8{data_in[160][0]}} & 8'hdc)^
({{8{data_in[160][1]}} & 8'h93)^
({{8{data_in[160][2]}} & 8'hd)^
({{8{data_in[160][3]}} & 8'h1a)^
({{8{data_in[160][4]}} & 8'h34)^
({{8{data_in[160][5]}} & 8'h68)^
({{8{data_in[160][6]}} & 8'hd0)^
({{8{data_in[160][7]}} & 8'h8b)^
({{8{data_in[161][0]}} & 8'had)^
({{8{data_in[161][1]}} & 8'h71)^
({{8{data_in[161][2]}} & 8'he2)^
({{8{data_in[161][3]}} & 8'hef)^
({{8{data_in[161][4]}} & 8'hf5)^
({{8{data_in[161][5]}} & 8'hc1)^
({{8{data_in[161][6]}} & 8'ha9)^
({{8{data_in[161][7]}} & 8'h79)^
({{8{data_in[162][0]}} & 8'h54)^
({{8{data_in[162][1]}} & 8'ha8)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[162][2]}} & 8'h7b)^
({{8{data_in[162][3]}} & 8'hf6)^
({{8{data_in[162][4]}} & 8'hc7)^
({{8{data_in[162][5]}} & 8'ha5)^
({{8{data_in[162][6]}} & 8'h61)^
({{8{data_in[162][7]}} & 8'hc2)^
({{8{data_in[163][0]}} & 8'hf5)^
({{8{data_in[163][1]}} & 8'hc1)^
({{8{data_in[163][2]}} & 8'ha9)^
({{8{data_in[163][3]}} & 8'h79)^
({{8{data_in[163][4]}} & 8'hf2)^
({{8{data_in[163][5]}} & 8'hcf)^
({{8{data_in[163][6]}} & 8'hb5)^
({{8{data_in[163][7]}} & 8'h41)^
({{8{data_in[164][0]}} & 8'h55)^
({{8{data_in[164][1]}} & 8'haa)^
({{8{data_in[164][2]}} & 8'h7f)^
({{8{data_in[164][3]}} & 8'hf6)^
({{8{data_in[164][4]}} & 8'hd7)^
({{8{data_in[164][5]}} & 8'h85)^
({{8{data_in[164][6]}} & 8'h21)^
({{8{data_in[164][7]}} & 8'h42)^
({{8{data_in[165][0]}} & 8'he6)^
({{8{data_in[165][1]}} & 8'he7)^
({{8{data_in[165][2]}} & 8'he5)^
({{8{data_in[165][3]}} & 8'he1)^
({{8{data_in[165][4]}} & 8'he9)^
({{8{data_in[165][5]}} & 8'hf9)^
({{8{data_in[165][6]}} & 8'hd9)^
({{8{data_in[165][7]}} & 8'h99)^
({{8{data_in[166][0]}} & 8'h1f)^
({{8{data_in[166][1]}} & 8'h3e)^
({{8{data_in[166][2]}} & 8'h7c)^
({{8{data_in[166][3]}} & 8'hf8)^
({{8{data_in[166][4]}} & 8'hdb)^
({{8{data_in[166][5]}} & 8'h9d)^
({{8{data_in[166][6]}} & 8'h11)^
({{8{data_in[166][7]}} & 8'h22)^
({{8{data_in[167][0]}} & 8'hb3)^
({{8{data_in[167][1]}} & 8'h4d)^
({{8{data_in[167][2]}} & 8'h9a)^
({{8{data_in[167][3]}} & 8'h1f)^
({{8{data_in[167][4]}} & 8'h3e)^
({{8{data_in[167][5]}} & 8'h7c)^
({{8{data_in[167][6]}} & 8'hf8)^
({{8{data_in[167][7]}} & 8'hdb)^
({{8{data_in[168][0]}} & 8'he1)^
({{8{data_in[168][1]}} & 8'he9)^
({{8{data_in[168][2]}} & 8'hf9)^
({{8{data_in[168][3]}} & 8'hd9)^
({{8{data_in[168][4]}} & 8'h99)^
({{8{data_in[168][5]}} & 8'h19)^
({{8{data_in[168][6]}} & 8'h32)^
({{8{data_in[168][7]}} & 8'h64)^
({{8{data_in[169][0]}} & 8'h37)^
({{8{data_in[169][1]}} & 8'h6e)^
({{8{data_in[169][2]}} & 8'hdc)^
({{8{data_in[169][3]}} & 8'h93)^
({{8{data_in[169][4]}} & 8'hd)^
({{8{data_in[169][5]}} & 8'h1a)^
({{8{data_in[169][6]}} & 8'h34)^
({{8{data_in[169][7]}} & 8'h68)^
({{8{data_in[170][0]}} & 8'he8)^
({{8{data_in[170][1]}} & 8'hfb)^
({{8{data_in[170][2]}} & 8'hdd)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[170][3]}} & 8'h91)^
({{8{data_in[170][4]}} & 8'h9)^
({{8{data_in[170][5]}} & 8'h12)^
({{8{data_in[170][6]}} & 8'h24)^
({{8{data_in[170][7]}} & 8'h48)^
({{8{data_in[171][0]}} & 8'h2b)^
({{8{data_in[171][1]}} & 8'h56)^
({{8{data_in[171][2]}} & 8'hac)^
({{8{data_in[171][3]}} & 8'h73)^
({{8{data_in[171][4]}} & 8/he6)^
({{8{data_in[171][5]}} & 8/he7)^
({{8{data_in[171][6]}} & 8/he5)^
({{8{data_in[171][7]}} & 8/he1)^
({{8{data_in[172][0]}} & 8'h54)^
({{8{data_in[172][1]}} & 8'ha8)^
({{8{data_in[172][2]}} & 8'h7b)^
({{8{data_in[172][3]}} & 8'hf6)^
({{8{data_in[172][4]}} & 8'hc7)^
({{8{data_in[172][5]}} & 8'ha5)^
({{8{data_in[172][6]}} & 8'h61)^
({{8{data_in[172][7]}} & 8'hc2)^
({{8{data_in[173][0]}} & 8'h80)^
({{8{data_in[173][1]}} & 8'h2b)^
({{8{data_in[173][2]}} & 8'h56)^
({{8{data_in[173][3]}} & 8'hac)^
({{8{data_in[173][4]}} & 8'h73)^
({{8{data_in[173][5]}} & 8/he6)^
({{8{data_in[173][6]}} & 8/he7)^
({{8{data_in[173][7]}} & 8/he5)^
({{8{data_in[174][0]}} & 8'h16)^
({{8{data_in[174][1]}} & 8'h2c)^
({{8{data_in[174][2]}} & 8'h58)^
({{8{data_in[174][3]}} & 8'hb0)^
({{8{data_in[174][4]}} & 8'h4b)^
({{8{data_in[174][5]}} & 8'h96)^
({{8{data_in[174][6]}} & 8'h7)^
({{8{data_in[174][7]}} & 8/he)^
({{8{data_in[175][0]}} & 8'hc2)^
({{8{data_in[175][1]}} & 8'haf)^
({{8{data_in[175][2]}} & 8'h75)^
({{8{data_in[175][3]}} & 8'hea)^
({{8{data_in[175][4]}} & 8'hff)^
({{8{data_in[175][5]}} & 8'hd5)^
({{8{data_in[175][6]}} & 8'h81)^
({{8{data_in[175][7]}} & 8'h29)^
({{8{data_in[176][0]}} & 8'h94)^
({{8{data_in[176][1]}} & 8'h3)^
({{8{data_in[176][2]}} & 8'h6)^
({{8{data_in[176][3]}} & 8'hc)^
({{8{data_in[176][4]}} & 8'h18)^
({{8{data_in[176][5]}} & 8'h30)^
({{8{data_in[176][6]}} & 8'h60)^
({{8{data_in[176][7]}} & 8'hc0)^
({{8{data_in[177][0]}} & 8'h55)^
({{8{data_in[177][1]}} & 8'haa)^
({{8{data_in[177][2]}} & 8'h7f)^
({{8{data_in[177][3]}} & 8'hfe)^
({{8{data_in[177][4]}} & 8'hd7)^
({{8{data_in[177][5]}} & 8'h85)^
({{8{data_in[177][6]}} & 8'h21)^
({{8{data_in[177][7]}} & 8'h42)^
({{8{data_in[178][0]}} & 8'h7b)^
({{8{data_in[178][1]}} & 8'hf6)^
({{8{data_in[178][2]}} & 8'hc7)^
({{8{data_in[178][3]}} & 8'ha5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[178][4]}} & 8'h61)^
({{8{data_in[178][5]}} & 8'hc2)^
({{8{data_in[178][6]}} & 8'haf)^
({{8{data_in[178][7]}} & 8'h75)^
({{8{data_in[179][0]}} & 8'hbd)^
({{8{data_in[179][1]}} & 8'h51)^
({{8{data_in[179][2]}} & 8'ha2)^
({{8{data_in[179][3]}} & 8'h6f)^
({{8{data_in[179][4]}} & 8'hde)^
({{8{data_in[179][5]}} & 8'h97)^
({{8{data_in[179][6]}} & 8'h5)^
({{8{data_in[179][7]}} & 8'ha)^
({{8{data_in[180][0]}} & 8'h9f)^
({{8{data_in[180][1]}} & 8'h15)^
({{8{data_in[180][2]}} & 8'h2a)^
({{8{data_in[180][3]}} & 8'h54)^
({{8{data_in[180][4]}} & 8'ha8)^
({{8{data_in[180][5]}} & 8'h7b)^
({{8{data_in[180][6]}} & 8'hf6)^
({{8{data_in[180][7]}} & 8'hc7)^
({{8{data_in[181][0]}} & 8'hde)^
({{8{data_in[181][1]}} & 8'h97)^
({{8{data_in[181][2]}} & 8'h5)^
({{8{data_in[181][3]}} & 8'ha)^
({{8{data_in[181][4]}} & 8'h14)^
({{8{data_in[181][5]}} & 8'h28)^
({{8{data_in[181][6]}} & 8'h50)^
({{8{data_in[181][7]}} & 8'ha0)^
({{8{data_in[182][0]}} & 8'hb1)^
({{8{data_in[182][1]}} & 8'h49)^
({{8{data_in[182][2]}} & 8'h92)^
({{8{data_in[182][3]}} & 8'hf)^
({{8{data_in[182][4]}} & 8'h1e)^
({{8{data_in[182][5]}} & 8'h3c)^
({{8{data_in[182][6]}} & 8'h78)^
({{8{data_in[182][7]}} & 8'hf0)^
({{8{data_in[183][0]}} & 8'hee)^
({{8{data_in[183][1]}} & 8'hf7)^
({{8{data_in[183][2]}} & 8'hc5)^
({{8{data_in[183][3]}} & 8'ha1)^
({{8{data_in[183][4]}} & 8'h69)^
({{8{data_in[183][5]}} & 8'hd2)^
({{8{data_in[183][6]}} & 8'h8f)^
({{8{data_in[183][7]}} & 8'h35)^
({{8{data_in[184][0]}} & 8'h1a)^
({{8{data_in[184][1]}} & 8'h34)^
({{8{data_in[184][2]}} & 8'h68)^
({{8{data_in[184][3]}} & 8'hd0)^
({{8{data_in[184][4]}} & 8'h8b)^
({{8{data_in[184][5]}} & 8'h3d)^
({{8{data_in[184][6]}} & 8'h7a)^
({{8{data_in[184][7]}} & 8'hf4)^
({{8{data_in[185][0]}} & 8'he0)^
({{8{data_in[185][1]}} & 8'heb)^
({{8{data_in[185][2]}} & 8'hfd)^
({{8{data_in[185][3]}} & 8'hd1)^
({{8{data_in[185][4]}} & 8'h89)^
({{8{data_in[185][5]}} & 8'h39)^
({{8{data_in[185][6]}} & 8'h72)^
({{8{data_in[185][7]}} & 8'he4)^
({{8{data_in[186][0]}} & 8'hd8)^
({{8{data_in[186][1]}} & 8'h9b)^
({{8{data_in[186][2]}} & 8'h1d)^
({{8{data_in[186][3]}} & 8'h3a)^
({{8{data_in[186][4]}} & 8'h74)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[186][5]}} & 8'he8)^
({{8{data_in[186][6]}} & 8'hfb)^
({{8{data_in[186][7]}} & 8'hdd)^
({{8{data_in[187][0]}} & 8'h1f)^
({{8{data_in[187][1]}} & 8'h3e)^
({{8{data_in[187][2]}} & 8'h7c)^
({{8{data_in[187][3]}} & 8'hf8)^
({{8{data_in[187][4]}} & 8'hdb)^
({{8{data_in[187][5]}} & 8'h9d)^
({{8{data_in[187][6]}} & 8'h11)^
({{8{data_in[187][7]}} & 8'h22)^
({{8{data_in[188][0]}} & 8'hb4)^
({{8{data_in[188][1]}} & 8'h43)^
({{8{data_in[188][2]}} & 8'h86)^
({{8{data_in[188][3]}} & 8'h27)^
({{8{data_in[188][4]}} & 8'h4e)^
({{8{data_in[188][5]}} & 8'h9c)^
({{8{data_in[188][6]}} & 8'h13)^
({{8{data_in[188][7]}} & 8'h26)^
({{8{data_in[189][0]}} & 8'hf5)^
({{8{data_in[189][1]}} & 8'hc1)^
({{8{data_in[189][2]}} & 8'ha9)^
({{8{data_in[189][3]}} & 8'h79)^
({{8{data_in[189][4]}} & 8'hf2)^
({{8{data_in[189][5]}} & 8'hcf)^
({{8{data_in[189][6]}} & 8'hb5)^
({{8{data_in[189][7]}} & 8'h41)^
({{8{data_in[190][0]}} & 8'haf)^
({{8{data_in[190][1]}} & 8'h75)^
({{8{data_in[190][2]}} & 8'hea)^
({{8{data_in[190][3]}} & 8'hff)^
({{8{data_in[190][4]}} & 8'hd5)^
({{8{data_in[190][5]}} & 8'h81)^
({{8{data_in[190][6]}} & 8'h29)^
({{8{data_in[190][7]}} & 8'h52)^
({{8{data_in[191][0]}} & 8'hfe)^
({{8{data_in[191][1]}} & 8'hd7)^
({{8{data_in[191][2]}} & 8'h85)^
({{8{data_in[191][3]}} & 8'h21)^
({{8{data_in[191][4]}} & 8'h42)^
({{8{data_in[191][5]}} & 8'h84)^
({{8{data_in[191][6]}} & 8'h23)^
({{8{data_in[191][7]}} & 8'h46)^
({{8{data_in[192][0]}} & 8'hb2)^
({{8{data_in[192][1]}} & 8'h4f)^
({{8{data_in[192][2]}} & 8'h9e)^
({{8{data_in[192][3]}} & 8'h17)^
({{8{data_in[192][4]}} & 8'h2e)^
({{8{data_in[192][5]}} & 8'h5c)^
({{8{data_in[192][6]}} & 8'hb8)^
({{8{data_in[192][7]}} & 8'h5b)^
({{8{data_in[193][0]}} & 8'h15)^
({{8{data_in[193][1]}} & 8'h2a)^
({{8{data_in[193][2]}} & 8'h54)^
({{8{data_in[193][3]}} & 8'ha8)^
({{8{data_in[193][4]}} & 8'h7b)^
({{8{data_in[193][5]}} & 8'hf6)^
({{8{data_in[193][6]}} & 8'hc7)^
({{8{data_in[193][7]}} & 8'ha5)^
({{8{data_in[194][0]}} & 8'h11)^
({{8{data_in[194][1]}} & 8'h22)^
({{8{data_in[194][2]}} & 8'h44)^
({{8{data_in[194][3]}} & 8'h88)^
({{8{data_in[194][4]}} & 8'h3b)^
({{8{data_in[194][5]}} & 8'h76)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[194][6]}} & 8'hec)^
({{8{data_in[194][7]}} & 8'hf3)^
({{8{data_in[195][0]}} & 8'had)^
({{8{data_in[195][1]}} & 8'h71)^
({{8{data_in[195][2]}} & 8'he2)^
({{8{data_in[195][3]}} & 8'hef)^
({{8{data_in[195][4]}} & 8'hf5)^
({{8{data_in[195][5]}} & 8'hc1)^
({{8{data_in[195][6]}} & 8'ha9)^
({{8{data_in[195][7]}} & 8'h79)^
({{8{data_in[196][0]}} & 8'hb6)^
({{8{data_in[196][1]}} & 8'h47)^
({{8{data_in[196][2]}} & 8'h8e)^
({{8{data_in[196][3]}} & 8'h37)^
({{8{data_in[196][4]}} & 8'h6e)^
({{8{data_in[196][5]}} & 8'hdc)^
({{8{data_in[196][6]}} & 8'h93)^
({{8{data_in[196][7]}} & 8'hd)^
({{8{data_in[197][0]}} & 8'h89)^
({{8{data_in[197][1]}} & 8'h39)^
({{8{data_in[197][2]}} & 8'h72)^
({{8{data_in[197][3]}} & 8'he4)^
({{8{data_in[197][4]}} & 8'he3)^
({{8{data_in[197][5]}} & 8'hed)^
({{8{data_in[197][6]}} & 8'hf1)^
({{8{data_in[197][7]}} & 8'hc9)^
({{8{data_in[198][0]}} & 8'h1f)^
({{8{data_in[198][1]}} & 8'h3e)^
({{8{data_in[198][2]}} & 8'h7c)^
({{8{data_in[198][3]}} & 8'hf8)^
({{8{data_in[198][4]}} & 8'hdb)^
({{8{data_in[198][5]}} & 8'h9d)^
({{8{data_in[198][6]}} & 8'h11)^
({{8{data_in[198][7]}} & 8'h22)^
({{8{data_in[199][0]}} & 8'h28)^
({{8{data_in[199][1]}} & 8'h50)^
({{8{data_in[199][2]}} & 8'ha0)^
({{8{data_in[199][3]}} & 8'h6b)^
({{8{data_in[199][4]}} & 8'hd6)^
({{8{data_in[199][5]}} & 8'h87)^
({{8{data_in[199][6]}} & 8'h25)^
({{8{data_in[199][7]}} & 8'h4a)^
({{8{data_in[200][0]}} & 8'h46)^
({{8{data_in[200][1]}} & 8'h8c)^
({{8{data_in[200][2]}} & 8'h33)^
({{8{data_in[200][3]}} & 8'h66)^
({{8{data_in[200][4]}} & 8'hcc)^
({{8{data_in[200][5]}} & 8'hb3)^
({{8{data_in[200][6]}} & 8'h4d)^
({{8{data_in[200][7]}} & 8'h9a)^
({{8{data_in[201][0]}} & 8'hf8)^
({{8{data_in[201][1]}} & 8'hdb)^
({{8{data_in[201][2]}} & 8'h9d)^
({{8{data_in[201][3]}} & 8'h11)^
({{8{data_in[201][4]}} & 8'h22)^
({{8{data_in[201][5]}} & 8'h44)^
({{8{data_in[201][6]}} & 8'h88)^
({{8{data_in[201][7]}} & 8'h3b)^
({{8{data_in[202][0]}} & 8'hb9)^
({{8{data_in[202][1]}} & 8'h59)^
({{8{data_in[202][2]}} & 8'hb2)^
({{8{data_in[202][3]}} & 8'h4f)^
({{8{data_in[202][4]}} & 8'h9e)^
({{8{data_in[202][5]}} & 8'h17)^
({{8{data_in[202][6]}} & 8'h2e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[202][7]}} & 8'h5c)^
({{8{data_in[203][0]}} & 8'ha8)^
({{8{data_in[203][1]}} & 8'h7b)^
({{8{data_in[203][2]}} & 8'hf6)^
({{8{data_in[203][3]}} & 8'hc7)^
({{8{data_in[203][4]}} & 8'ha5)^
({{8{data_in[203][5]}} & 8'h61)^
({{8{data_in[203][6]}} & 8'hc2)^
({{8{data_in[203][7]}} & 8'haf)^
({{8{data_in[204][0]}} & 8'h9a)^
({{8{data_in[204][1]}} & 8'h1f)^
({{8{data_in[204][2]}} & 8'h3e)^
({{8{data_in[204][3]}} & 8'h7c)^
({{8{data_in[204][4]}} & 8'hf8)^
({{8{data_in[204][5]}} & 8'hdb)^
({{8{data_in[204][6]}} & 8'h9d)^
({{8{data_in[204][7]}} & 8'h11)^
({{8{data_in[205][0]}} & 8'h78)^
({{8{data_in[205][1]}} & 8'hf0)^
({{8{data_in[205][2]}} & 8'hcb)^
({{8{data_in[205][3]}} & 8'hbd)^
({{8{data_in[205][4]}} & 8'h51)^
({{8{data_in[205][5]}} & 8'ha2)^
({{8{data_in[205][6]}} & 8'h6f)^
({{8{data_in[205][7]}} & 8'hde)^
({{8{data_in[206][0]}} & 8'h84)^
({{8{data_in[206][1]}} & 8'h23)^
({{8{data_in[206][2]}} & 8'h46)^
({{8{data_in[206][3]}} & 8'h8c)^
({{8{data_in[206][4]}} & 8'h33)^
({{8{data_in[206][5]}} & 8'h66)^
({{8{data_in[206][6]}} & 8'hcc)^
({{8{data_in[206][7]}} & 8'hb3)^
({{8{data_in[207][0]}} & 8'h79)^
({{8{data_in[207][1]}} & 8'hf2)^
({{8{data_in[207][2]}} & 8'hcf)^
({{8{data_in[207][3]}} & 8'hb5)^
({{8{data_in[207][4]}} & 8'h41)^
({{8{data_in[207][5]}} & 8'h82)^
({{8{data_in[207][6]}} & 8'h2f)^
({{8{data_in[207][7]}} & 8'h5e)^
({{8{data_in[208][0]}} & 8'h5f)^
({{8{data_in[208][1]}} & 8'hbe)^
({{8{data_in[208][2]}} & 8'h57)^
({{8{data_in[208][3]}} & 8'hae)^
({{8{data_in[208][4]}} & 8'h77)^
({{8{data_in[208][5]}} & 8'hee)^
({{8{data_in[208][6]}} & 8'hf7)^
({{8{data_in[208][7]}} & 8'hc5)^
({{8{data_in[209][0]}} & 8'h9e)^
({{8{data_in[209][1]}} & 8'h17)^
({{8{data_in[209][2]}} & 8'h2e)^
({{8{data_in[209][3]}} & 8'h5c)^
({{8{data_in[209][4]}} & 8'hb8)^
({{8{data_in[209][5]}} & 8'h5b)^
({{8{data_in[209][6]}} & 8'hb6)^
({{8{data_in[209][7]}} & 8'h47)^
({{8{data_in[210][0]}} & 8'h14)^
({{8{data_in[210][1]}} & 8'h28)^
({{8{data_in[210][2]}} & 8'h50)^
({{8{data_in[210][3]}} & 8'ha0)^
({{8{data_in[210][4]}} & 8'h6b)^
({{8{data_in[210][5]}} & 8'hd6)^
({{8{data_in[210][6]}} & 8'h87)^
({{8{data_in[210][7]}} & 8'h25)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[211][0]}} & 8'hdc)^
({{8{data_in[211][1]}} & 8'h93)^
({{8{data_in[211][2]}} & 8'hd)^
({{8{data_in[211][3]}} & 8'h1a)^
({{8{data_in[211][4]}} & 8'h34)^
({{8{data_in[211][5]}} & 8'h68)^
({{8{data_in[211][6]}} & 8'hd0)^
({{8{data_in[211][7]}} & 8'h8b)^
({{8{data_in[212][0]}} & 8'he0)^
({{8{data_in[212][1]}} & 8'heb)^
({{8{data_in[212][2]}} & 8'hfd)^
({{8{data_in[212][3]}} & 8'hd1)^
({{8{data_in[212][4]}} & 8'h89)^
({{8{data_in[212][5]}} & 8'h39)^
({{8{data_in[212][6]}} & 8'h72)^
({{8{data_in[212][7]}} & 8'he4)^
({{8{data_in[213][0]}} & 8'h94)^
({{8{data_in[213][1]}} & 8'h3)^
({{8{data_in[213][2]}} & 8'h6)^
({{8{data_in[213][3]}} & 8'hc)^
({{8{data_in[213][4]}} & 8'h18)^
({{8{data_in[213][5]}} & 8'h30)^
({{8{data_in[213][6]}} & 8'h60)^
({{8{data_in[213][7]}} & 8'hc0)^
({{8{data_in[214][0]}} & 8'h3b)^
({{8{data_in[214][1]}} & 8'h76)^
({{8{data_in[214][2]}} & 8'hec)^
({{8{data_in[214][3]}} & 8'hf3)^
({{8{data_in[214][4]}} & 8'hcd)^
({{8{data_in[214][5]}} & 8'hb1)^
({{8{data_in[214][6]}} & 8'h49)^
({{8{data_in[214][7]}} & 8'h92)^
({{8{data_in[215][0]}} & 8'h46)^
({{8{data_in[215][1]}} & 8'h8c)^
({{8{data_in[215][2]}} & 8'h33)^
({{8{data_in[215][3]}} & 8'h66)^
({{8{data_in[215][4]}} & 8'hcc)^
({{8{data_in[215][5]}} & 8'hb3)^
({{8{data_in[215][6]}} & 8'h4d)^
({{8{data_in[215][7]}} & 8'h9a)^
({{8{data_in[216][0]}} & 8'h8e)^
({{8{data_in[216][1]}} & 8'h37)^
({{8{data_in[216][2]}} & 8'h6e)^
({{8{data_in[216][3]}} & 8'hdc)^
({{8{data_in[216][4]}} & 8'h93)^
({{8{data_in[216][5]}} & 8'hd)^
({{8{data_in[216][6]}} & 8'h1a)^
({{8{data_in[216][7]}} & 8'h34)^
({{8{data_in[217][0]}} & 8'h54)^
({{8{data_in[217][1]}} & 8'ha8)^
({{8{data_in[217][2]}} & 8'h7b)^
({{8{data_in[217][3]}} & 8'hf6)^
({{8{data_in[217][4]}} & 8'hc7)^
({{8{data_in[217][5]}} & 8'ha5)^
({{8{data_in[217][6]}} & 8'h61)^
({{8{data_in[217][7]}} & 8'hc2)^
({{8{data_in[218][0]}} & 8'h50)^
({{8{data_in[218][1]}} & 8'ha0)^
({{8{data_in[218][2]}} & 8'h6b)^
({{8{data_in[218][3]}} & 8'hd6)^
({{8{data_in[218][4]}} & 8'h87)^
({{8{data_in[218][5]}} & 8'h25)^
({{8{data_in[218][6]}} & 8'h4a)^
({{8{data_in[218][7]}} & 8'h94)^
({{8{data_in[219][0]}} & 8'h8c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[219][1]}} & 8'h33)^
({{8{data_in[219][2]}} & 8'h66)^
({{8{data_in[219][3]}} & 8'hcc)^
({{8{data_in[219][4]}} & 8'hb3)^
({{8{data_in[219][5]}} & 8'h4d)^
({{8{data_in[219][6]}} & 8'h9a)^
({{8{data_in[219][7]}} & 8'h1f)^
({{8{data_in[220][0]}} & 8'h2f)^
({{8{data_in[220][1]}} & 8'h5e)^
({{8{data_in[220][2]}} & 8'hbc)^
({{8{data_in[220][3]}} & 8'h53)^
({{8{data_in[220][4]}} & 8'ha6)^
({{8{data_in[220][5]}} & 8'h67)^
({{8{data_in[220][6]}} & 8'hce)^
({{8{data_in[220][7]}} & 8'hb7)^
({{8{data_in[221][0]}} & 8'ha0)^
({{8{data_in[221][1]}} & 8'h6b)^
({{8{data_in[221][2]}} & 8'hd6)^
({{8{data_in[221][3]}} & 8'h87)^
({{8{data_in[221][4]}} & 8'h25)^
({{8{data_in[221][5]}} & 8'h4a)^
({{8{data_in[221][6]}} & 8'h94)^
({{8{data_in[221][7]}} & 8'h3)^
({{8{data_in[222][0]}} & 8'h57)^
({{8{data_in[222][1]}} & 8'hae)^
({{8{data_in[222][2]}} & 8'h77)^
({{8{data_in[222][3]}} & 8'hee)^
({{8{data_in[222][4]}} & 8'hf7)^
({{8{data_in[222][5]}} & 8'hc5)^
({{8{data_in[222][6]}} & 8'ha1)^
({{8{data_in[222][7]}} & 8'h69)^
({{8{data_in[223][0]}} & 8'hd3)^
({{8{data_in[223][1]}} & 8'h8d)^
({{8{data_in[223][2]}} & 8'h31)^
({{8{data_in[223][3]}} & 8'h62)^
({{8{data_in[223][4]}} & 8'hc4)^
({{8{data_in[223][5]}} & 8'ha3)^
({{8{data_in[223][6]}} & 8'h6d)^
({{8{data_in[223][7]}} & 8'hda)^
({{8{data_in[224][0]}} & 8'h3d)^
({{8{data_in[224][1]}} & 8'h7a)^
({{8{data_in[224][2]}} & 8'hf4)^
({{8{data_in[224][3]}} & 8'hc3)^
({{8{data_in[224][4]}} & 8'had)^
({{8{data_in[224][5]}} & 8'h71)^
({{8{data_in[224][6]}} & 8'he2)^
({{8{data_in[224][7]}} & 8'hef)^
({{8{data_in[225][0]}} & 8'hcd)^
({{8{data_in[225][1]}} & 8'hb1)^
({{8{data_in[225][2]}} & 8'h49)^
({{8{data_in[225][3]}} & 8'h92)^
({{8{data_in[225][4]}} & 8'hf)^
({{8{data_in[225][5]}} & 8'h1e)^
({{8{data_in[225][6]}} & 8'h3c)^
({{8{data_in[225][7]}} & 8'h78)^
({{8{data_in[226][0]}} & 8'h5b)^
({{8{data_in[226][1]}} & 8'hb6)^
({{8{data_in[226][2]}} & 8'h47)^
({{8{data_in[226][3]}} & 8'h8e)^
({{8{data_in[226][4]}} & 8'h37)^
({{8{data_in[226][5]}} & 8'h6e)^
({{8{data_in[226][6]}} & 8'hdc)^
({{8{data_in[226][7]}} & 8'h93)^
({{8{data_in[227][0]}} & 8'h28)^
({{8{data_in[227][1]}} & 8'h50)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[227][2]}} & 8'ha0)^
({{8{data_in[227][3]}} & 8'h6b)^
({{8{data_in[227][4]}} & 8'hd6)^
({{8{data_in[227][5]}} & 8'h87)^
({{8{data_in[227][6]}} & 8'h25)^
({{8{data_in[227][7]}} & 8'h4a)^
({{8{data_in[228][0]}} & 8'had)^
({{8{data_in[228][1]}} & 8'h71)^
({{8{data_in[228][2]}} & 8'he2)^
({{8{data_in[228][3]}} & 8'hef)^
({{8{data_in[228][4]}} & 8'hf5)^
({{8{data_in[228][5]}} & 8'hc1)^
({{8{data_in[228][6]}} & 8'ha9)^
({{8{data_in[228][7]}} & 8'h79)^
({{8{data_in[229][0]}} & 8'hd7)^
({{8{data_in[229][1]}} & 8'h85)^
({{8{data_in[229][2]}} & 8'h21)^
({{8{data_in[229][3]}} & 8'h42)^
({{8{data_in[229][4]}} & 8'h84)^
({{8{data_in[229][5]}} & 8'h23)^
({{8{data_in[229][6]}} & 8'h46)^
({{8{data_in[229][7]}} & 8'h8c)^
({{8{data_in[230][0]}} & 8'hee)^
({{8{data_in[230][1]}} & 8'hf7)^
({{8{data_in[230][2]}} & 8'hc5)^
({{8{data_in[230][3]}} & 8'ha1)^
({{8{data_in[230][4]}} & 8'h69)^
({{8{data_in[230][5]}} & 8'hd2)^
({{8{data_in[230][6]}} & 8'h8f)^
({{8{data_in[230][7]}} & 8'h35)^
({{8{data_in[231][0]}} & 8'h91)^
({{8{data_in[231][1]}} & 8'h9)^
({{8{data_in[231][2]}} & 8'h12)^
({{8{data_in[231][3]}} & 8'h24)^
({{8{data_in[231][4]}} & 8'h48)^
({{8{data_in[231][5]}} & 8'h90)^
({{8{data_in[231][6]}} & 8'hb)^
({{8{data_in[231][7]}} & 8'h16)^
({{8{data_in[232][0]}} & 8'h4c)^
({{8{data_in[232][1]}} & 8'h98)^
({{8{data_in[232][2]}} & 8'h1b)^
({{8{data_in[232][3]}} & 8'h36)^
({{8{data_in[232][4]}} & 8'h6c)^
({{8{data_in[232][5]}} & 8'hd8)^
({{8{data_in[232][6]}} & 8'h9b)^
({{8{data_in[232][7]}} & 8'h1d)^
({{8{data_in[233][0]}} & 8'h4f)^
({{8{data_in[233][1]}} & 8'h9e)^
({{8{data_in[233][2]}} & 8'h17)^
({{8{data_in[233][3]}} & 8'h2e)^
({{8{data_in[233][4]}} & 8'h5c)^
({{8{data_in[233][5]}} & 8'hb8)^
({{8{data_in[233][6]}} & 8'h5b)^
({{8{data_in[233][7]}} & 8'hb6)^
({{8{data_in[234][0]}} & 8'hf7)^
({{8{data_in[234][1]}} & 8'hc5)^
({{8{data_in[234][2]}} & 8'ha1)^
({{8{data_in[234][3]}} & 8'h69)^
({{8{data_in[234][4]}} & 8'hd2)^
({{8{data_in[234][5]}} & 8'h8f)^
({{8{data_in[234][6]}} & 8'h35)^
({{8{data_in[234][7]}} & 8'h6a)^
({{8{data_in[235][0]}} & 8'h18)^
({{8{data_in[235][1]}} & 8'h30)^
({{8{data_in[235][2]}} & 8'h60)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[235][3]}} & 8'hc0)^
({{8{data_in[235][4]}} & 8'hab)^
({{8{data_in[235][5]}} & 8'h7d)^
({{8{data_in[235][6]}} & 8'hfa)^
({{8{data_in[235][7]}} & 8'hdf)^
({{8{data_in[236][0]}} & 8'h99)^
({{8{data_in[236][1]}} & 8'h19)^
({{8{data_in[236][2]}} & 8'h32)^
({{8{data_in[236][3]}} & 8'h64)^
({{8{data_in[236][4]}} & 8'hc8)^
({{8{data_in[236][5]}} & 8'hbb)^
({{8{data_in[236][6]}} & 8'h5d)^
({{8{data_in[236][7]}} & 8'hba)^
({{8{data_in[237][0]}} & 8'h28)^
({{8{data_in[237][1]}} & 8'h50)^
({{8{data_in[237][2]}} & 8'ha0)^
({{8{data_in[237][3]}} & 8'h6b)^
({{8{data_in[237][4]}} & 8'hd6)^
({{8{data_in[237][5]}} & 8'h87)^
({{8{data_in[237][6]}} & 8'h25)^
({{8{data_in[237][7]}} & 8'h4a)^
({{8{data_in[238][0]}} & 8'hd0)^
({{8{data_in[238][1]}} & 8'h8b)^
({{8{data_in[238][2]}} & 8'h3d)^
({{8{data_in[238][3]}} & 8'h7a)^
({{8{data_in[238][4]}} & 8'hf4)^
({{8{data_in[238][5]}} & 8'hc3)^
({{8{data_in[238][6]}} & 8'had)^
({{8{data_in[238][7]}} & 8'h71)^
({{8{data_in[239][0]}} & 8'h58)^
({{8{data_in[239][1]}} & 8'hb0)^
({{8{data_in[239][2]}} & 8'h4b)^
({{8{data_in[239][3]}} & 8'h96)^
({{8{data_in[239][4]}} & 8'h7)^
({{8{data_in[239][5]}} & 8'he)^
({{8{data_in[239][6]}} & 8'h1c)^
({{8{data_in[239][7]}} & 8'h38)^
({{8{data_in[240][0]}} & 8'h9)^
({{8{data_in[240][1]}} & 8'h12)^
({{8{data_in[240][2]}} & 8'h24)^
({{8{data_in[240][3]}} & 8'h48)^
({{8{data_in[240][4]}} & 8'h90)^
({{8{data_in[240][5]}} & 8'hb)^
({{8{data_in[240][6]}} & 8'h16)^
({{8{data_in[240][7]}} & 8'h2c)^
({{8{data_in[241][0]}} & 8'h68)^
({{8{data_in[241][1]}} & 8'hd0)^
({{8{data_in[241][2]}} & 8'h8b)^
({{8{data_in[241][3]}} & 8'h3d)^
({{8{data_in[241][4]}} & 8'h7a)^
({{8{data_in[241][5]}} & 8'hf4)^
({{8{data_in[241][6]}} & 8'hc3)^
({{8{data_in[241][7]}} & 8'had);

data_out[247] = ({8{data_in[0][0]}} & 8'ha7)^
    ({8{data_in[0][1]}} & 8'h65)^
    ({8{data_in[0][2]}} & 8'hca)^
    ({8{data_in[0][3]}} & 8'hbf)^
    ({8{data_in[0][4]}} & 8'h55)^
    ({8{data_in[0][5]}} & 8'haa)^
    ({8{data_in[0][6]}} & 8'h7f)^
    ({8{data_in[0][7]}} & 8'hfe)^
    ({8{data_in[1][0]}} & 8'hdb)^
    ({8{data_in[1][1]}} & 8'h9d)^
    ({8{data_in[1][2]}} & 8'h11)^
    ({8{data_in[1][3]}} & 8'h22)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[1][4]}} & 8'h44)^
({{8{data_in[1][5]}} & 8'h88)^
({{8{data_in[1][6]}} & 8'h3b)^
({{8{data_in[1][7]}} & 8'h76)^
({{8{data_in[2][0]}} & 8'h37)^
({{8{data_in[2][1]}} & 8'h6e)^
({{8{data_in[2][2]}} & 8'hdcc)^
({{8{data_in[2][3]}} & 8'h93)^
({{8{data_in[2][4]}} & 8'hd)^
({{8{data_in[2][5]}} & 8'h1a)^
({{8{data_in[2][6]}} & 8'h34)^
({{8{data_in[2][7]}} & 8'h68)^
({{8{data_in[3][0]}} & 8'hca)^
({{8{data_in[3][1]}} & 8'hbf)^
({{8{data_in[3][2]}} & 8'h55)^
({{8{data_in[3][3]}} & 8'haa)^
({{8{data_in[3][4]}} & 8'h7f)^
({{8{data_in[3][5]}} & 8'hfe)^
({{8{data_in[3][6]}} & 8'hd7)^
({{8{data_in[3][7]}} & 8'h85)^
({{8{data_in[4][0]}} & 8'h9e)^
({{8{data_in[4][1]}} & 8'h17)^
({{8{data_in[4][2]}} & 8'h2e)^
({{8{data_in[4][3]}} & 8'h5c)^
({{8{data_in[4][4]}} & 8'hb8)^
({{8{data_in[4][5]}} & 8'h5b)^
({{8{data_in[4][6]}} & 8'hb6)^
({{8{data_in[4][7]}} & 8'h47)^
({{8{data_in[5][0]}} & 8'h27)^
({{8{data_in[5][1]}} & 8'h4e)^
({{8{data_in[5][2]}} & 8'h9c)^
({{8{data_in[5][3]}} & 8'h13)^
({{8{data_in[5][4]}} & 8'h26)^
({{8{data_in[5][5]}} & 8'h4c)^
({{8{data_in[5][6]}} & 8'h98)^
({{8{data_in[5][7]}} & 8'h1b)^
({{8{data_in[6][0]}} & 8'h8b)^
({{8{data_in[6][1]}} & 8'h3d)^
({{8{data_in[6][2]}} & 8'h7a)^
({{8{data_in[6][3]}} & 8'hf4)^
({{8{data_in[6][4]}} & 8'hc3)^
({{8{data_in[6][5]}} & 8'had)^
({{8{data_in[6][6]}} & 8'h71)^
({{8{data_in[6][7]}} & 8'he2)^
({{8{data_in[7][0]}} & 8'h28)^
({{8{data_in[7][1]}} & 8'h50)^
({{8{data_in[7][2]}} & 8'ha0)^
({{8{data_in[7][3]}} & 8'h6b)^
({{8{data_in[7][4]}} & 8'hd6)^
({{8{data_in[7][5]}} & 8'h87)^
({{8{data_in[7][6]}} & 8'h25)^
({{8{data_in[7][7]}} & 8'h4a)^
({{8{data_in[8][0]}} & 8'h15)^
({{8{data_in[8][1]}} & 8'h2a)^
({{8{data_in[8][2]}} & 8'h54)^
({{8{data_in[8][3]}} & 8'ha8)^
({{8{data_in[8][4]}} & 8'h7b)^
({{8{data_in[8][5]}} & 8'hf6)^
({{8{data_in[8][6]}} & 8'hc7)^
({{8{data_in[8][7]}} & 8'ha5)^
({{8{data_in[9][0]}} & 8'h86)^
({{8{data_in[9][1]}} & 8'h27)^
({{8{data_in[9][2]}} & 8'h4e)^
({{8{data_in[9][3]}} & 8'h9c)^
({{8{data_in[9][4]}} & 8'h13)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[9][5]}} & 8'h26)^
({{8{data_in[9][6]}} & 8'h4c)^
({{8{data_in[9][7]}} & 8'h98)^
({{8{data_in[10][0]}} & 8'h17)^
({{8{data_in[10][1]}} & 8'h2e)^
({{8{data_in[10][2]}} & 8'h5c)^
({{8{data_in[10][3]}} & 8'hb8)^
({{8{data_in[10][4]}} & 8'h5b)^
({{8{data_in[10][5]}} & 8'hb6)^
({{8{data_in[10][6]}} & 8'h47)^
({{8{data_in[10][7]}} & 8'h8e)^
({{8{data_in[11][0]}} & 8'h90)^
({{8{data_in[11][1]}} & 8'hb)^
({{8{data_in[11][2]}} & 8'h16)^
({{8{data_in[11][3]}} & 8'h2c)^
({{8{data_in[11][4]}} & 8'h58)^
({{8{data_in[11][5]}} & 8'hb0)^
({{8{data_in[11][6]}} & 8'h4b)^
({{8{data_in[11][7]}} & 8'h96)^
({{8{data_in[12][0]}} & 8'h86)^
({{8{data_in[12][1]}} & 8'h27)^
({{8{data_in[12][2]}} & 8'h4e)^
({{8{data_in[12][3]}} & 8'h9c)^
({{8{data_in[12][4]}} & 8'h13)^
({{8{data_in[12][5]}} & 8'h26)^
({{8{data_in[12][6]}} & 8'h4c)^
({{8{data_in[12][7]}} & 8'h98)^
({{8{data_in[13][0]}} & 8'h2c)^
({{8{data_in[13][1]}} & 8'h58)^
({{8{data_in[13][2]}} & 8'hb0)^
({{8{data_in[13][3]}} & 8'h4b)^
({{8{data_in[13][4]}} & 8'h96)^
({{8{data_in[13][5]}} & 8'h7)^
({{8{data_in[13][6]}} & 8'he)^
({{8{data_in[13][7]}} & 8'h1c)^
({{8{data_in[14][0]}} & 8'hb)^
({{8{data_in[14][1]}} & 8'h16)^
({{8{data_in[14][2]}} & 8'h2c)^
({{8{data_in[14][3]}} & 8'h58)^
({{8{data_in[14][4]}} & 8'hb0)^
({{8{data_in[14][5]}} & 8'h4b)^
({{8{data_in[14][6]}} & 8'h96)^
({{8{data_in[14][7]}} & 8'h7)^
({{8{data_in[15][0]}} & 8'h7)^
({{8{data_in[15][1]}} & 8'he)^
({{8{data_in[15][2]}} & 8'h1c)^
({{8{data_in[15][3]}} & 8'h38)^
({{8{data_in[15][4]}} & 8'h70)^
({{8{data_in[15][5]}} & 8'he0)^
({{8{data_in[15][6]}} & 8'heb)^
({{8{data_in[15][7]}} & 8'hfd)^
({{8{data_in[16][0]}} & 8'h5f)^
({{8{data_in[16][1]}} & 8'hbe)^
({{8{data_in[16][2]}} & 8'h57)^
({{8{data_in[16][3]}} & 8'hae)^
({{8{data_in[16][4]}} & 8'h77)^
({{8{data_in[16][5]}} & 8'hee)^
({{8{data_in[16][6]}} & 8'hf7)^
({{8{data_in[16][7]}} & 8'hc5)^
({{8{data_in[17][0]}} & 8'h49)^
({{8{data_in[17][1]}} & 8'h92)^
({{8{data_in[17][2]}} & 8'hf)^
({{8{data_in[17][3]}} & 8'h1e)^
({{8{data_in[17][4]}} & 8'h3c)^
({{8{data_in[17][5]}} & 8'h78)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[17][6]}} & 8'hf0)^
({{8{data_in[17][7]}} & 8'hcb)^
({{8{data_in[18][0]}} & 8'h6)^
({{8{data_in[18][1]}} & 8'hc)^
({{8{data_in[18][2]}} & 8'h18)^
({{8{data_in[18][3]}} & 8'h30)^
({{8{data_in[18][4]}} & 8'h60)^
({{8{data_in[18][5]}} & 8'hc0)^
({{8{data_in[18][6]}} & 8'hab)^
({{8{data_in[18][7]}} & 8'h7d)^
({{8{data_in[19][0]}} & 8'h63)^
({{8{data_in[19][1]}} & 8'hc6)^
({{8{data_in[19][2]}} & 8'ha7)^
({{8{data_in[19][3]}} & 8'h65)^
({{8{data_in[19][4]}} & 8'hca)^
({{8{data_in[19][5]}} & 8'hbf)^
({{8{data_in[19][6]}} & 8'h55)^
({{8{data_in[19][7]}} & 8'haa)^
({{8{data_in[20][0]}} & 8'h7f)^
({{8{data_in[20][1]}} & 8'hfe)^
({{8{data_in[20][2]}} & 8'hd7)^
({{8{data_in[20][3]}} & 8'h85)^
({{8{data_in[20][4]}} & 8'h21)^
({{8{data_in[20][5]}} & 8'h42)^
({{8{data_in[20][6]}} & 8'h84)^
({{8{data_in[20][7]}} & 8'h23)^
({{8{data_in[21][0]}} & 8'h5e)^
({{8{data_in[21][1]}} & 8'hbc)^
({{8{data_in[21][2]}} & 8'h53)^
({{8{data_in[21][3]}} & 8'ha6)^
({{8{data_in[21][4]}} & 8'h67)^
({{8{data_in[21][5]}} & 8'hce)^
({{8{data_in[21][6]}} & 8'hb7)^
({{8{data_in[21][7]}} & 8'h45)^
({{8{data_in[22][0]}} & 8'h84)^
({{8{data_in[22][1]}} & 8'h23)^
({{8{data_in[22][2]}} & 8'h46)^
({{8{data_in[22][3]}} & 8'h8c)^
({{8{data_in[22][4]}} & 8'h33)^
({{8{data_in[22][5]}} & 8'h66)^
({{8{data_in[22][6]}} & 8'hcc)^
({{8{data_in[22][7]}} & 8'hb3)^
({{8{data_in[23][0]}} & 8'hf5)^
({{8{data_in[23][1]}} & 8'hc1)^
({{8{data_in[23][2]}} & 8'ha9)^
({{8{data_in[23][3]}} & 8'h79)^
({{8{data_in[23][4]}} & 8'hf2)^
({{8{data_in[23][5]}} & 8'hcf)^
({{8{data_in[23][6]}} & 8'hb5)^
({{8{data_in[23][7]}} & 8'h41)^
({{8{data_in[24][0]}} & 8'ha4)^
({{8{data_in[24][1]}} & 8'h63)^
({{8{data_in[24][2]}} & 8'hc6)^
({{8{data_in[24][3]}} & 8'ha7)^
({{8{data_in[24][4]}} & 8'h65)^
({{8{data_in[24][5]}} & 8'hca)^
({{8{data_in[24][6]}} & 8'hbf)^
({{8{data_in[24][7]}} & 8'h55)^
({{8{data_in[25][0]}} & 8'h6f)^
({{8{data_in[25][1]}} & 8'hde)^
({{8{data_in[25][2]}} & 8'h97)^
({{8{data_in[25][3]}} & 8'h5)^
({{8{data_in[25][4]}} & 8'ha)^
({{8{data_in[25][5]}} & 8'h14)^
({{8{data_in[25][6]}} & 8'h28)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[25][7]}} & 8'h50)^
({{8{data_in[26][0]}} & 8'h72)^
({{8{data_in[26][1]}} & 8'he4)^
({{8{data_in[26][2]}} & 8'he3)^
({{8{data_in[26][3]}} & 8'hed)^
({{8{data_in[26][4]}} & 8'hf1)^
({{8{data_in[26][5]}} & 8'hc9)^
({{8{data_in[26][6]}} & 8'hb9)^
({{8{data_in[26][7]}} & 8'h59)^
({{8{data_in[27][0]}} & 8'h28)^
({{8{data_in[27][1]}} & 8'h50)^
({{8{data_in[27][2]}} & 8'ha0)^
({{8{data_in[27][3]}} & 8'h6b)^
({{8{data_in[27][4]}} & 8'hd6)^
({{8{data_in[27][5]}} & 8'h87)^
({{8{data_in[27][6]}} & 8'h25)^
({{8{data_in[27][7]}} & 8'h4a)^
({{8{data_in[28][0]}} & 8'h6f)^
({{8{data_in[28][1]}} & 8'hde)^
({{8{data_in[28][2]}} & 8'h97)^
({{8{data_in[28][3]}} & 8'h5)^
({{8{data_in[28][4]}} & 8'ha)^
({{8{data_in[28][5]}} & 8'h14)^
({{8{data_in[28][6]}} & 8'h28)^
({{8{data_in[28][7]}} & 8'h50)^
({{8{data_in[29][0]}} & 8'he)^
({{8{data_in[29][1]}} & 8'h1c)^
({{8{data_in[29][2]}} & 8'h38)^
({{8{data_in[29][3]}} & 8'h70)^
({{8{data_in[29][4]}} & 8'he0)^
({{8{data_in[29][5]}} & 8'heb)^
({{8{data_in[29][6]}} & 8'hfd)^
({{8{data_in[29][7]}} & 8'hd1)^
({{8{data_in[30][0]}} & 8'h97)^
({{8{data_in[30][1]}} & 8'h5)^
({{8{data_in[30][2]}} & 8'ha)^
({{8{data_in[30][3]}} & 8'h14)^
({{8{data_in[30][4]}} & 8'h28)^
({{8{data_in[30][5]}} & 8'h50)^
({{8{data_in[30][6]}} & 8'ha0)^
({{8{data_in[30][7]}} & 8'h6b)^
({{8{data_in[31][0]}} & 8'h60)^
({{8{data_in[31][1]}} & 8'hc0)^
({{8{data_in[31][2]}} & 8'hab)^
({{8{data_in[31][3]}} & 8'h7d)^
({{8{data_in[31][4]}} & 8'hfa)^
({{8{data_in[31][5]}} & 8'hdf)^
({{8{data_in[31][6]}} & 8'h95)^
({{8{data_in[31][7]}} & 8'h1)^
({{8{data_in[32][0]}} & 8'hae)^
({{8{data_in[32][1]}} & 8'h77)^
({{8{data_in[32][2]}} & 8'hee)^
({{8{data_in[32][3]}} & 8'hf7)^
({{8{data_in[32][4]}} & 8'hc5)^
({{8{data_in[32][5]}} & 8'ha1)^
({{8{data_in[32][6]}} & 8'h69)^
({{8{data_in[32][7]}} & 8'hd2)^
({{8{data_in[33][0]}} & 8'h16)^
({{8{data_in[33][1]}} & 8'h2c)^
({{8{data_in[33][2]}} & 8'h58)^
({{8{data_in[33][3]}} & 8'hb0)^
({{8{data_in[33][4]}} & 8'h4b)^
({{8{data_in[33][5]}} & 8'h96)^
({{8{data_in[33][6]}} & 8'h7)^
({{8{data_in[33][7]}} & 8'he)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[34][0]}} & 8'ha8)^
({{8{data_in[34][1]}} & 8'h7b)^
({{8{data_in[34][2]}} & 8'hf6)^
({{8{data_in[34][3]}} & 8'hc7)^
({{8{data_in[34][4]}} & 8'ha5)^
({{8{data_in[34][5]}} & 8'h61)^
({{8{data_in[34][6]}} & 8'hc2)^
({{8{data_in[34][7]}} & 8'haf)^
({{8{data_in[35][0]}} & 8'h61)^
({{8{data_in[35][1]}} & 8'hc2)^
({{8{data_in[35][2]}} & 8'haf)^
({{8{data_in[35][3]}} & 8'h75)^
({{8{data_in[35][4]}} & 8'hea)^
({{8{data_in[35][5]}} & 8'hff)^
({{8{data_in[35][6]}} & 8'hd5)^
({{8{data_in[35][7]}} & 8'h81)^
({{8{data_in[36][0]}} & 8'h69)^
({{8{data_in[36][1]}} & 8'hd2)^
({{8{data_in[36][2]}} & 8'h8f)^
({{8{data_in[36][3]}} & 8'h35)^
({{8{data_in[36][4]}} & 8'h6a)^
({{8{data_in[36][5]}} & 8'hd4)^
({{8{data_in[36][6]}} & 8'h83)^
({{8{data_in[36][7]}} & 8'h2d)^
({{8{data_in[37][0]}} & 8'h77)^
({{8{data_in[37][1]}} & 8'hee)^
({{8{data_in[37][2]}} & 8'hf7)^
({{8{data_in[37][3]}} & 8'hc5)^
({{8{data_in[37][4]}} & 8'ha1)^
({{8{data_in[37][5]}} & 8'h69)^
({{8{data_in[37][6]}} & 8'hd2)^
({{8{data_in[37][7]}} & 8'h8f)^
({{8{data_in[38][0]}} & 8'hbd)^
({{8{data_in[38][1]}} & 8'h51)^
({{8{data_in[38][2]}} & 8'ha2)^
({{8{data_in[38][3]}} & 8'h6f)^
({{8{data_in[38][4]}} & 8'hde)^
({{8{data_in[38][5]}} & 8'h97)^
({{8{data_in[38][6]}} & 8'h5)^
({{8{data_in[38][7]}} & 8'ha)^
({{8{data_in[39][0]}} & 8'h4c)^
({{8{data_in[39][1]}} & 8'h98)^
({{8{data_in[39][2]}} & 8'h1b)^
({{8{data_in[39][3]}} & 8'h36)^
({{8{data_in[39][4]}} & 8'h6c)^
({{8{data_in[39][5]}} & 8'hd8)^
({{8{data_in[39][6]}} & 8'h9b)^
({{8{data_in[39][7]}} & 8'h1d)^
({{8{data_in[40][0]}} & 8'hc0)^
({{8{data_in[40][1]}} & 8'hab)^
({{8{data_in[40][2]}} & 8'h7d)^
({{8{data_in[40][3]}} & 8'hfa)^
({{8{data_in[40][4]}} & 8'hdf)^
({{8{data_in[40][5]}} & 8'h95)^
({{8{data_in[40][6]}} & 8'h1)^
({{8{data_in[40][7]}} & 8'h2)^
({{8{data_in[41][0]}} & 8'h72)^
({{8{data_in[41][1]}} & 8'he4)^
({{8{data_in[41][2]}} & 8'he3)^
({{8{data_in[41][3]}} & 8'hed)^
({{8{data_in[41][4]}} & 8'hf1)^
({{8{data_in[41][5]}} & 8'hc9)^
({{8{data_in[41][6]}} & 8'hb9)^
({{8{data_in[41][7]}} & 8'h59)^
({{8{data_in[42][0]}} & 8'h3a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[42][1]}} & 8'h74)^
({{8{data_in[42][2]}} & 8'he8)^
({{8{data_in[42][3]}} & 8'hfb)^
({{8{data_in[42][4]}} & 8'hdd)^
({{8{data_in[42][5]}} & 8'h91)^
({{8{data_in[42][6]}} & 8'h9)^
({{8{data_in[42][7]}} & 8'h12)^
({{8{data_in[43][0]}} & 8'ha1)^
({{8{data_in[43][1]}} & 8'h69)^
({{8{data_in[43][2]}} & 8'hd2)^
({{8{data_in[43][3]}} & 8'h8f)^
({{8{data_in[43][4]}} & 8'h35)^
({{8{data_in[43][5]}} & 8'h6a)^
({{8{data_in[43][6]}} & 8'hd4)^
({{8{data_in[43][7]}} & 8'h83)^
({{8{data_in[44][0]}} & 8'h3)^
({{8{data_in[44][1]}} & 8'h6)^
({{8{data_in[44][2]}} & 8'hc)^
({{8{data_in[44][3]}} & 8'h18)^
({{8{data_in[44][4]}} & 8'h30)^
({{8{data_in[44][5]}} & 8'h60)^
({{8{data_in[44][6]}} & 8'hc0)^
({{8{data_in[44][7]}} & 8'hab)^
({{8{data_in[45][0]}} & 8'h26)^
({{8{data_in[45][1]}} & 8'h4c)^
({{8{data_in[45][2]}} & 8'h98)^
({{8{data_in[45][3]}} & 8'h1b)^
({{8{data_in[45][4]}} & 8'h36)^
({{8{data_in[45][5]}} & 8'h6c)^
({{8{data_in[45][6]}} & 8'hd8)^
({{8{data_in[45][7]}} & 8'h9b)^
({{8{data_in[46][0]}} & 8'h79)^
({{8{data_in[46][1]}} & 8'hf2)^
({{8{data_in[46][2]}} & 8'hcf)^
({{8{data_in[46][3]}} & 8'hb5)^
({{8{data_in[46][4]}} & 8'h41)^
({{8{data_in[46][5]}} & 8'h82)^
({{8{data_in[46][6]}} & 8'h2f)^
({{8{data_in[46][7]}} & 8'h5e)^
({{8{data_in[47][0]}} & 8'hd2)^
({{8{data_in[47][1]}} & 8'h8f)^
({{8{data_in[47][2]}} & 8'h35)^
({{8{data_in[47][3]}} & 8'h6a)^
({{8{data_in[47][4]}} & 8'hd4)^
({{8{data_in[47][5]}} & 8'h83)^
({{8{data_in[47][6]}} & 8'h2d)^
({{8{data_in[47][7]}} & 8'h5a)^
({{8{data_in[48][0]}} & 8'h1a)^
({{8{data_in[48][1]}} & 8'h34)^
({{8{data_in[48][2]}} & 8'h68)^
({{8{data_in[48][3]}} & 8'hd0)^
({{8{data_in[48][4]}} & 8'h8b)^
({{8{data_in[48][5]}} & 8'h3d)^
({{8{data_in[48][6]}} & 8'h7a)^
({{8{data_in[48][7]}} & 8'hf4)^
({{8{data_in[49][0]}} & 8'h7a)^
({{8{data_in[49][1]}} & 8'hf4)^
({{8{data_in[49][2]}} & 8'hc3)^
({{8{data_in[49][3]}} & 8'had)^
({{8{data_in[49][4]}} & 8'h71)^
({{8{data_in[49][5]}} & 8'he2)^
({{8{data_in[49][6]}} & 8'hef)^
({{8{data_in[49][7]}} & 8'hf5)^
({{8{data_in[50][0]}} & 8'h53)^
({{8{data_in[50][1]}} & 8'ha6})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[50][2]}} & 8'h67)^
({{8{data_in[50][3]}} & 8'hce)^
({{8{data_in[50][4]}} & 8'hb7)^
({{8{data_in[50][5]}} & 8'h45)^
({{8{data_in[50][6]}} & 8'h8a)^
({{8{data_in[50][7]}} & 8'h3f)^
({{8{data_in[51][0]}} & 8'h3a)^
({{8{data_in[51][1]}} & 8'h74)^
({{8{data_in[51][2]}} & 8'he8)^
({{8{data_in[51][3]}} & 8'hfb)^
({{8{data_in[51][4]}} & 8'hdd)^
({{8{data_in[51][5]}} & 8'h91)^
({{8{data_in[51][6]}} & 8'h9)^
({{8{data_in[51][7]}} & 8'h12)^
({{8{data_in[52][0]}} & 8'h7b)^
({{8{data_in[52][1]}} & 8'hf6)^
({{8{data_in[52][2]}} & 8'hc7)^
({{8{data_in[52][3]}} & 8'ha5)^
({{8{data_in[52][4]}} & 8'h61)^
({{8{data_in[52][5]}} & 8'hc2)^
({{8{data_in[52][6]}} & 8'haf)^
({{8{data_in[52][7]}} & 8'h75)^
({{8{data_in[53][0]}} & 8'h51)^
({{8{data_in[53][1]}} & 8'ha2)^
({{8{data_in[53][2]}} & 8'h6f)^
({{8{data_in[53][3]}} & 8'hde)^
({{8{data_in[53][4]}} & 8'h97)^
({{8{data_in[53][5]}} & 8'h5)^
({{8{data_in[53][6]}} & 8'ha)^
({{8{data_in[53][7]}} & 8'h14)^
({{8{data_in[54][0]}} & 8'h7b)^
({{8{data_in[54][1]}} & 8'hf6)^
({{8{data_in[54][2]}} & 8'hc7)^
({{8{data_in[54][3]}} & 8'ha5)^
({{8{data_in[54][4]}} & 8'h61)^
({{8{data_in[54][5]}} & 8'hc2)^
({{8{data_in[54][6]}} & 8'haf)^
({{8{data_in[54][7]}} & 8'h75)^
({{8{data_in[55][0]}} & 8'hc7)^
({{8{data_in[55][1]}} & 8'ha5)^
({{8{data_in[55][2]}} & 8'h61)^
({{8{data_in[55][3]}} & 8'hc2)^
({{8{data_in[55][4]}} & 8'haf)^
({{8{data_in[55][5]}} & 8'h75)^
({{8{data_in[55][6]}} & 8'hea)^
({{8{data_in[55][7]}} & 8'hff)^
({{8{data_in[56][0]}} & 8'hdf)^
({{8{data_in[56][1]}} & 8'h95)^
({{8{data_in[56][2]}} & 8'h1)^
({{8{data_in[56][3]}} & 8'h2)^
({{8{data_in[56][4]}} & 8'h4)^
({{8{data_in[56][5]}} & 8'h8)^
({{8{data_in[56][6]}} & 8'h10)^
({{8{data_in[56][7]}} & 8'h20)^
({{8{data_in[57][0]}} & 8'hc)^
({{8{data_in[57][1]}} & 8'h18)^
({{8{data_in[57][2]}} & 8'h30)^
({{8{data_in[57][3]}} & 8'h60)^
({{8{data_in[57][4]}} & 8'hc0)^
({{8{data_in[57][5]}} & 8'hab)^
({{8{data_in[57][6]}} & 8'h7d)^
({{8{data_in[57][7]}} & 8'hfa)^
({{8{data_in[58][0]}} & 8'hb2)^
({{8{data_in[58][1]}} & 8'h4f)^
({{8{data_in[58][2]}} & 8'h9e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[58][3]}} & 8'h17)^
({{8{data_in[58][4]}} & 8'h2e)^
({{8{data_in[58][5]}} & 8'h5c)^
({{8{data_in[58][6]}} & 8'hb8)^
({{8{data_in[58][7]}} & 8'h5b)^
({{8{data_in[59][0]}} & 8'hed)^
({{8{data_in[59][1]}} & 8'hf1)^
({{8{data_in[59][2]}} & 8'hc9)^
({{8{data_in[59][3]}} & 8'hb9)^
({{8{data_in[59][4]}} & 8'h59)^
({{8{data_in[59][5]}} & 8'hb2)^
({{8{data_in[59][6]}} & 8'h4f)^
({{8{data_in[59][7]}} & 8'h9e)^
({{8{data_in[60][0]}} & 8'h51)^
({{8{data_in[60][1]}} & 8'ha2)^
({{8{data_in[60][2]}} & 8'h6f)^
({{8{data_in[60][3]}} & 8'hde)^
({{8{data_in[60][4]}} & 8'h97)^
({{8{data_in[60][5]}} & 8'h5)^
({{8{data_in[60][6]}} & 8'ha)^
({{8{data_in[60][7]}} & 8'h14)^
({{8{data_in[61][0]}} & 8'hbb)^
({{8{data_in[61][1]}} & 8'h5d)^
({{8{data_in[61][2]}} & 8'hba)^
({{8{data_in[61][3]}} & 8'h5f)^
({{8{data_in[61][4]}} & 8'hbe)^
({{8{data_in[61][5]}} & 8'h57)^
({{8{data_in[61][6]}} & 8'hae)^
({{8{data_in[61][7]}} & 8'h77)^
({{8{data_in[62][0]}} & 8'h4a)^
({{8{data_in[62][1]}} & 8'h94)^
({{8{data_in[62][2]}} & 8'h3)^
({{8{data_in[62][3]}} & 8'h6)^
({{8{data_in[62][4]}} & 8'hc)^
({{8{data_in[62][5]}} & 8'h18)^
({{8{data_in[62][6]}} & 8'h30)^
({{8{data_in[62][7]}} & 8'h60)^
({{8{data_in[63][0]}} & 8'h1f)^
({{8{data_in[63][1]}} & 8'h3e)^
({{8{data_in[63][2]}} & 8'h7c)^
({{8{data_in[63][3]}} & 8'hf8)^
({{8{data_in[63][4]}} & 8'hdb)^
({{8{data_in[63][5]}} & 8'h9d)^
({{8{data_in[63][6]}} & 8'h11)^
({{8{data_in[63][7]}} & 8'h22)^
({{8{data_in[64][0]}} & 8'ha6)^
({{8{data_in[64][1]}} & 8'h67)^
({{8{data_in[64][2]}} & 8'hce)^
({{8{data_in[64][3]}} & 8'hb7)^
({{8{data_in[64][4]}} & 8'h45)^
({{8{data_in[64][5]}} & 8'h8a)^
({{8{data_in[64][6]}} & 8'h3f)^
({{8{data_in[64][7]}} & 8'h7e)^
({{8{data_in[65][0]}} & 8'hba)^
({{8{data_in[65][1]}} & 8'h5f)^
({{8{data_in[65][2]}} & 8'hbe)^
({{8{data_in[65][3]}} & 8'h57)^
({{8{data_in[65][4]}} & 8'hae)^
({{8{data_in[65][5]}} & 8'h77)^
({{8{data_in[65][6]}} & 8'hee)^
({{8{data_in[65][7]}} & 8'hf7)^
({{8{data_in[66][0]}} & 8'hff)^
({{8{data_in[66][1]}} & 8'hd5)^
({{8{data_in[66][2]}} & 8'h81)^
({{8{data_in[66][3]}} & 8'h29})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[66][4]}} & 8'h52)^
({{8{data_in[66][5]}} & 8'ha4)^
({{8{data_in[66][6]}} & 8'h63)^
({{8{data_in[66][7]}} & 8'hc6)^
({{8{data_in[67][0]}} & 8'ha8)^
({{8{data_in[67][1]}} & 8'h7b)^
({{8{data_in[67][2]}} & 8'hf6)^
({{8{data_in[67][3]}} & 8'hc7)^
({{8{data_in[67][4]}} & 8'ha5)^
({{8{data_in[67][5]}} & 8'h61)^
({{8{data_in[67][6]}} & 8'hc2)^
({{8{data_in[67][7]}} & 8'haf)^
({{8{data_in[68][0]}} & 8'hb2)^
({{8{data_in[68][1]}} & 8'h4f)^
({{8{data_in[68][2]}} & 8'h9e)^
({{8{data_in[68][3]}} & 8'h17)^
({{8{data_in[68][4]}} & 8'h2e)^
({{8{data_in[68][5]}} & 8'h5c)^
({{8{data_in[68][6]}} & 8'hb8)^
({{8{data_in[68][7]}} & 8'h5b)^
({{8{data_in[69][0]}} & 8'ha8)^
({{8{data_in[69][1]}} & 8'h7b)^
({{8{data_in[69][2]}} & 8'hf6)^
({{8{data_in[69][3]}} & 8'hc7)^
({{8{data_in[69][4]}} & 8'ha5)^
({{8{data_in[69][5]}} & 8'h61)^
({{8{data_in[69][6]}} & 8'hc2)^
({{8{data_in[69][7]}} & 8'haf)^
({{8{data_in[70][0]}} & 8'he0)^
({{8{data_in[70][1]}} & 8'heb)^
({{8{data_in[70][2]}} & 8'hfd)^
({{8{data_in[70][3]}} & 8'hd1)^
({{8{data_in[70][4]}} & 8'h89)^
({{8{data_in[70][5]}} & 8'h39)^
({{8{data_in[70][6]}} & 8'h72)^
({{8{data_in[70][7]}} & 8'he4)^
({{8{data_in[71][0]}} & 8'h17)^
({{8{data_in[71][1]}} & 8'h2e)^
({{8{data_in[71][2]}} & 8'h5c)^
({{8{data_in[71][3]}} & 8'hb8)^
({{8{data_in[71][4]}} & 8'h5b)^
({{8{data_in[71][5]}} & 8'hb6)^
({{8{data_in[71][6]}} & 8'h47)^
({{8{data_in[71][7]}} & 8'h8e)^
({{8{data_in[72][0]}} & 8'h94)^
({{8{data_in[72][1]}} & 8'h3)^
({{8{data_in[72][2]}} & 8'h6)^
({{8{data_in[72][3]}} & 8'hc)^
({{8{data_in[72][4]}} & 8'h18)^
({{8{data_in[72][5]}} & 8'h30)^
({{8{data_in[72][6]}} & 8'h60)^
({{8{data_in[72][7]}} & 8'hc0)^
({{8{data_in[73][0]}} & 8'h40)^
({{8{data_in[73][1]}} & 8'h80)^
({{8{data_in[73][2]}} & 8'h2b)^
({{8{data_in[73][3]}} & 8'h56)^
({{8{data_in[73][4]}} & 8'hac)^
({{8{data_in[73][5]}} & 8'h73)^
({{8{data_in[73][6]}} & 8'he6)^
({{8{data_in[73][7]}} & 8'he7)^
({{8{data_in[74][0]}} & 8'h51)^
({{8{data_in[74][1]}} & 8'ha2)^
({{8{data_in[74][2]}} & 8'h6f)^
({{8{data_in[74][3]}} & 8'hde)^
({{8{data_in[74][4]}} & 8'h97})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[74][5]}} & 8'h5)^
({{8{data_in[74][6]}} & 8'ha)^
({{8{data_in[74][7]}} & 8'h14)^
({{8{data_in[75][0]}} & 8'h5)^
({{8{data_in[75][1]}} & 8'ha)^
({{8{data_in[75][2]}} & 8'h14)^
({{8{data_in[75][3]}} & 8'h28)^
({{8{data_in[75][4]}} & 8'h50)^
({{8{data_in[75][5]}} & 8'ha0)^
({{8{data_in[75][6]}} & 8'h6b)^
({{8{data_in[75][7]}} & 8'hd6)^
({{8{data_in[76][0]}} & 8'h32)^
({{8{data_in[76][1]}} & 8'h64)^
({{8{data_in[76][2]}} & 8'hc8)^
({{8{data_in[76][3]}} & 8'hbb)^
({{8{data_in[76][4]}} & 8'h5d)^
({{8{data_in[76][5]}} & 8'hba)^
({{8{data_in[76][6]}} & 8'h5f)^
({{8{data_in[76][7]}} & 8'hbe)^
({{8{data_in[77][0]}} & 8'h59)^
({{8{data_in[77][1]}} & 8'hb2)^
({{8{data_in[77][2]}} & 8'h4f)^
({{8{data_in[77][3]}} & 8'h9e)^
({{8{data_in[77][4]}} & 8'h17)^
({{8{data_in[77][5]}} & 8'h2e)^
({{8{data_in[77][6]}} & 8'h5c)^
({{8{data_in[77][7]}} & 8'hb8)^
({{8{data_in[78][0]}} & 8'h47)^
({{8{data_in[78][1]}} & 8'h8e)^
({{8{data_in[78][2]}} & 8'h37)^
({{8{data_in[78][3]}} & 8'h6e)^
({{8{data_in[78][4]}} & 8'hdc)^
({{8{data_in[78][5]}} & 8'h93)^
({{8{data_in[78][6]}} & 8'hd)^
({{8{data_in[78][7]}} & 8'h1a)^
({{8{data_in[79][0]}} & 8'h5d)^
({{8{data_in[79][1]}} & 8'hba)^
({{8{data_in[79][2]}} & 8'h5f)^
({{8{data_in[79][3]}} & 8'hbe)^
({{8{data_in[79][4]}} & 8'h57)^
({{8{data_in[79][5]}} & 8'hae)^
({{8{data_in[79][6]}} & 8'h77)^
({{8{data_in[79][7]}} & 8'hee)^
({{8{data_in[80][0]}} & 8'h6c)^
({{8{data_in[80][1]}} & 8'hd8)^
({{8{data_in[80][2]}} & 8'h9b)^
({{8{data_in[80][3]}} & 8'h1d)^
({{8{data_in[80][4]}} & 8'h3a)^
({{8{data_in[80][5]}} & 8'h74)^
({{8{data_in[80][6]}} & 8'he8)^
({{8{data_in[80][7]}} & 8'hfb)^
({{8{data_in[81][0]}} & 8'he7)^
({{8{data_in[81][1]}} & 8'he5)^
({{8{data_in[81][2]}} & 8'he1)^
({{8{data_in[81][3]}} & 8'he9)^
({{8{data_in[81][4]}} & 8'hf9)^
({{8{data_in[81][5]}} & 8'hd9)^
({{8{data_in[81][6]}} & 8'h99)^
({{8{data_in[81][7]}} & 8'h19)^
({{8{data_in[82][0]}} & 8'h96)^
({{8{data_in[82][1]}} & 8'h7)^
({{8{data_in[82][2]}} & 8'he)^
({{8{data_in[82][3]}} & 8'h1c)^
({{8{data_in[82][4]}} & 8'h38)^
({{8{data_in[82][5]}} & 8'h70)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[82][6]}} & 8'h e0)^
({{8{data_in[82][7]}} & 8'h eb)^
({{8{data_in[83][0]}} & 8'h 44)^
({{8{data_in[83][1]}} & 8'h 88)^
({{8{data_in[83][2]}} & 8'h 3b)^
({{8{data_in[83][3]}} & 8'h 76)^
({{8{data_in[83][4]}} & 8'h ec)^
({{8{data_in[83][5]}} & 8'h f3)^
({{8{data_in[83][6]}} & 8'h cd)^
({{8{data_in[83][7]}} & 8'h b1)^
({{8{data_in[84][0]}} & 8'h 53)^
({{8{data_in[84][1]}} & 8'h a6)^
({{8{data_in[84][2]}} & 8'h 67)^
({{8{data_in[84][3]}} & 8'h ce)^
({{8{data_in[84][4]}} & 8'h b7)^
({{8{data_in[84][5]}} & 8'h 45)^
({{8{data_in[84][6]}} & 8'h 8a)^
({{8{data_in[84][7]}} & 8'h 3f)^
({{8{data_in[85][0]}} & 8'h 6b)^
({{8{data_in[85][1]}} & 8'h d6)^
({{8{data_in[85][2]}} & 8'h 87)^
({{8{data_in[85][3]}} & 8'h 25)^
({{8{data_in[85][4]}} & 8'h 4a)^
({{8{data_in[85][5]}} & 8'h 94)^
({{8{data_in[85][6]}} & 8'h 3)^
({{8{data_in[85][7]}} & 8'h 6)^
({{8{data_in[86][0]}} & 8'h 8f)^
({{8{data_in[86][1]}} & 8'h 35)^
({{8{data_in[86][2]}} & 8'h 6a)^
({{8{data_in[86][3]}} & 8'h d4)^
({{8{data_in[86][4]}} & 8'h 83)^
({{8{data_in[86][5]}} & 8'h 2d)^
({{8{data_in[86][6]}} & 8'h 5a)^
({{8{data_in[86][7]}} & 8'h b4)^
({{8{data_in[87][0]}} & 8'h 8c)^
({{8{data_in[87][1]}} & 8'h 33)^
({{8{data_in[87][2]}} & 8'h 66)^
({{8{data_in[87][3]}} & 8'h cc)^
({{8{data_in[87][4]}} & 8'h b3)^
({{8{data_in[87][5]}} & 8'h 4d)^
({{8{data_in[87][6]}} & 8'h 9a)^
({{8{data_in[87][7]}} & 8'h 1f)^
({{8{data_in[88][0]}} & 8'h 33)^
({{8{data_in[88][1]}} & 8'h 66)^
({{8{data_in[88][2]}} & 8'h cc)^
({{8{data_in[88][3]}} & 8'h b3)^
({{8{data_in[88][4]}} & 8'h 4d)^
({{8{data_in[88][5]}} & 8'h 9a)^
({{8{data_in[88][6]}} & 8'h 1f)^
({{8{data_in[88][7]}} & 8'h 3e)^
({{8{data_in[89][0]}} & 8'h 30)^
({{8{data_in[89][1]}} & 8'h 60)^
({{8{data_in[89][2]}} & 8'h c0)^
({{8{data_in[89][3]}} & 8'h ab)^
({{8{data_in[89][4]}} & 8'h 7d)^
({{8{data_in[89][5]}} & 8'h fa)^
({{8{data_in[89][6]}} & 8'h df)^
({{8{data_in[89][7]}} & 8'h 95)^
({{8{data_in[90][0]}} & 8'h 48)^
({{8{data_in[90][1]}} & 8'h 90)^
({{8{data_in[90][2]}} & 8'h b)^
({{8{data_in[90][3]}} & 8'h 16)^
({{8{data_in[90][4]}} & 8'h 2c)^
({{8{data_in[90][5]}} & 8'h 58)^
({{8{data_in[90][6]}} & 8'h b0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[90][7]}} & 8'h4b)^
({{8{data_in[91][0]}} & 8'h18)^
({{8{data_in[91][1]}} & 8'h30)^
({{8{data_in[91][2]}} & 8'h60)^
({{8{data_in[91][3]}} & 8'hc0)^
({{8{data_in[91][4]}} & 8'hab)^
({{8{data_in[91][5]}} & 8'h7d)^
({{8{data_in[91][6]}} & 8'hfa)^
({{8{data_in[91][7]}} & 8'hdf)^
({{8{data_in[92][0]}} & 8'he8)^
({{8{data_in[92][1]}} & 8'hfb)^
({{8{data_in[92][2]}} & 8'hdd)^
({{8{data_in[92][3]}} & 8'h91)^
({{8{data_in[92][4]}} & 8'h9)^
({{8{data_in[92][5]}} & 8'h12)^
({{8{data_in[92][6]}} & 8'h24)^
({{8{data_in[92][7]}} & 8'h48)^
({{8{data_in[93][0]}} & 8'haa)^
({{8{data_in[93][1]}} & 8'h7f)^
({{8{data_in[93][2]}} & 8'hfe)^
({{8{data_in[93][3]}} & 8'hd7)^
({{8{data_in[93][4]}} & 8'h85)^
({{8{data_in[93][5]}} & 8'h21)^
({{8{data_in[93][6]}} & 8'h42)^
({{8{data_in[93][7]}} & 8'h84)^
({{8{data_in[94][0]}} & 8'hd0)^
({{8{data_in[94][1]}} & 8'h8b)^
({{8{data_in[94][2]}} & 8'h3d)^
({{8{data_in[94][3]}} & 8'h7a)^
({{8{data_in[94][4]}} & 8'hf4)^
({{8{data_in[94][5]}} & 8'hc3)^
({{8{data_in[94][6]}} & 8'had)^
({{8{data_in[94][7]}} & 8'h71)^
({{8{data_in[95][0]}} & 8'h3f)^
({{8{data_in[95][1]}} & 8'h7e)^
({{8{data_in[95][2]}} & 8'hfc)^
({{8{data_in[95][3]}} & 8'hd3)^
({{8{data_in[95][4]}} & 8'h8d)^
({{8{data_in[95][5]}} & 8'h31)^
({{8{data_in[95][6]}} & 8'h62)^
({{8{data_in[95][7]}} & 8'hc4)^
({{8{data_in[96][0]}} & 8'h90)^
({{8{data_in[96][1]}} & 8'hb)^
({{8{data_in[96][2]}} & 8'h16)^
({{8{data_in[96][3]}} & 8'h2c)^
({{8{data_in[96][4]}} & 8'h58)^
({{8{data_in[96][5]}} & 8'hb0)^
({{8{data_in[96][6]}} & 8'h4b)^
({{8{data_in[96][7]}} & 8'h96)^
({{8{data_in[97][0]}} & 8'h1b)^
({{8{data_in[97][1]}} & 8'h36)^
({{8{data_in[97][2]}} & 8'h6c)^
({{8{data_in[97][3]}} & 8'hd8)^
({{8{data_in[97][4]}} & 8'h9b)^
({{8{data_in[97][5]}} & 8'h1d)^
({{8{data_in[97][6]}} & 8'h3a)^
({{8{data_in[97][7]}} & 8'h74)^
({{8{data_in[98][0]}} & 8'h24)^
({{8{data_in[98][1]}} & 8'h48)^
({{8{data_in[98][2]}} & 8'h90)^
({{8{data_in[98][3]}} & 8'hb)^
({{8{data_in[98][4]}} & 8'h16)^
({{8{data_in[98][5]}} & 8'h2c)^
({{8{data_in[98][6]}} & 8'h58)^
({{8{data_in[98][7]}} & 8'hb0})

```

Base 6.4 vs Base 6.3

```

({{8{data_in[99][0]}} & 8'h73)^
({{8{data_in[99][1]}} & 8'he6)^
({{8{data_in[99][2]}} & 8'he7)^
({{8{data_in[99][3]}} & 8'he5)^
({{8{data_in[99][4]}} & 8'he1)^
({{8{data_in[99][5]}} & 8'he9)^
({{8{data_in[99][6]}} & 8'hf9)^
({{8{data_in[99][7]}} & 8'hd9)^
({{8{data_in[100][0]}} & 8'h93)^
({{8{data_in[100][1]}} & 8'hd)^
({{8{data_in[100][2]}} & 8'h1a)^
({{8{data_in[100][3]}} & 8'h34)^
({{8{data_in[100][4]}} & 8'h68)^
({{8{data_in[100][5]}} & 8'hd0)^
({{8{data_in[100][6]}} & 8'h8b)^
({{8{data_in[100][7]}} & 8'h3d)^
({{8{data_in[101][0]}} & 8'h8b)^
({{8{data_in[101][1]}} & 8'h3d)^
({{8{data_in[101][2]}} & 8'h7a)^
({{8{data_in[101][3]}} & 8'hf4)^
({{8{data_in[101][4]}} & 8'hc3)^
({{8{data_in[101][5]}} & 8'had)^
({{8{data_in[101][6]}} & 8'h71)^
({{8{data_in[101][7]}} & 8'he2)^
({{8{data_in[102][0]}} & 8'haa)^
({{8{data_in[102][1]}} & 8'h7f)^
({{8{data_in[102][2]}} & 8'hfe)^
({{8{data_in[102][3]}} & 8'hd7)^
({{8{data_in[102][4]}} & 8'h85)^
({{8{data_in[102][5]}} & 8'h21)^
({{8{data_in[102][6]}} & 8'h42)^
({{8{data_in[102][7]}} & 8'h84)^
({{8{data_in[103][0]}} & 8'h37)^
({{8{data_in[103][1]}} & 8'h6e)^
({{8{data_in[103][2]}} & 8'hdc)^
({{8{data_in[103][3]}} & 8'h93)^
({{8{data_in[103][4]}} & 8'hd)^
({{8{data_in[103][5]}} & 8'h1a)^
({{8{data_in[103][6]}} & 8'h34)^
({{8{data_in[103][7]}} & 8'h68)^
({{8{data_in[104][0]}} & 8'hfc)^
({{8{data_in[104][1]}} & 8'hd3)^
({{8{data_in[104][2]}} & 8'h8d)^
({{8{data_in[104][3]}} & 8'h31)^
({{8{data_in[104][4]}} & 8'h62)^
({{8{data_in[104][5]}} & 8'hc4)^
({{8{data_in[104][6]}} & 8'ha3)^
({{8{data_in[104][7]}} & 8'h6d)^
({{8{data_in[105][0]}} & 8'h1d)^
({{8{data_in[105][1]}} & 8'h3a)^
({{8{data_in[105][2]}} & 8'h74)^
({{8{data_in[105][3]}} & 8'he8)^
({{8{data_in[105][4]}} & 8'hfb)^
({{8{data_in[105][5]}} & 8'hdd)^
({{8{data_in[105][6]}} & 8'h91)^
({{8{data_in[105][7]}} & 8'h9)^
({{8{data_in[106][0]}} & 8'h9f)^
({{8{data_in[106][1]}} & 8'h15)^
({{8{data_in[106][2]}} & 8'h2a)^
({{8{data_in[106][3]}} & 8'h54)^
({{8{data_in[106][4]}} & 8'ha8)^
({{8{data_in[106][5]}} & 8'h7b)^
({{8{data_in[106][6]}} & 8'hf6)^
({{8{data_in[106][7]}} & 8'hc7)^
({{8{data_in[107][0]}} & 8'ha1)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[107][1]}} & 8'h69)^
({{8{data_in[107][2]}} & 8'hd2)^
({{8{data_in[107][3]}} & 8'h8f)^
({{8{data_in[107][4]}} & 8'h35)^
({{8{data_in[107][5]}} & 8'h6a)^
({{8{data_in[107][6]}} & 8'hd4)^
({{8{data_in[107][7]}} & 8'h83)^
({{8{data_in[108][0]}} & 8'h89)^
({{8{data_in[108][1]}} & 8'h39)^
({{8{data_in[108][2]}} & 8'h72)^
({{8{data_in[108][3]}} & 8'he4)^
({{8{data_in[108][4]}} & 8'he3)^
({{8{data_in[108][5]}} & 8'hed)^
({{8{data_in[108][6]}} & 8'hf1)^
({{8{data_in[108][7]}} & 8'hc9)^
({{8{data_in[109][0]}} & 8'h7d)^
({{8{data_in[109][1]}} & 8'hfa)^
({{8{data_in[109][2]}} & 8'hdf)^
({{8{data_in[109][3]}} & 8'h95)^
({{8{data_in[109][4]}} & 8'h1)^
({{8{data_in[109][5]}} & 8'h2)^
({{8{data_in[109][6]}} & 8'h4)^
({{8{data_in[109][7]}} & 8'h8)^
({{8{data_in[110][0]}} & 8'hc0)^
({{8{data_in[110][1]}} & 8'hab)^
({{8{data_in[110][2]}} & 8'h7d)^
({{8{data_in[110][3]}} & 8'hfa)^
({{8{data_in[110][4]}} & 8'hdf)^
({{8{data_in[110][5]}} & 8'h95)^
({{8{data_in[110][6]}} & 8'h1)^
({{8{data_in[110][7]}} & 8'h2)^
({{8{data_in[111][0]}} & 8'h60)^
({{8{data_in[111][1]}} & 8'hc0)^
({{8{data_in[111][2]}} & 8'hab)^
({{8{data_in[111][3]}} & 8'h7d)^
({{8{data_in[111][4]}} & 8'hfa)^
({{8{data_in[111][5]}} & 8'hdf)^
({{8{data_in[111][6]}} & 8'h95)^
({{8{data_in[111][7]}} & 8'h1)^
({{8{data_in[112][0]}} & 8'h7e)^
({{8{data_in[112][1]}} & 8'hfc)^
({{8{data_in[112][2]}} & 8'hd3)^
({{8{data_in[112][3]}} & 8'h8d)^
({{8{data_in[112][4]}} & 8'h31)^
({{8{data_in[112][5]}} & 8'h62)^
({{8{data_in[112][6]}} & 8'hc4)^
({{8{data_in[112][7]}} & 8'ha3)^
({{8{data_in[113][0]}} & 8'h22)^
({{8{data_in[113][1]}} & 8'h44)^
({{8{data_in[113][2]}} & 8'h88)^
({{8{data_in[113][3]}} & 8'h3b)^
({{8{data_in[113][4]}} & 8'h76)^
({{8{data_in[113][5]}} & 8'hec)^
({{8{data_in[113][6]}} & 8'hf3)^
({{8{data_in[113][7]}} & 8'hcd)^
({{8{data_in[114][0]}} & 8'he4)^
({{8{data_in[114][1]}} & 8'he3)^
({{8{data_in[114][2]}} & 8'hed)^
({{8{data_in[114][3]}} & 8'hf1)^
({{8{data_in[114][4]}} & 8'hc9)^
({{8{data_in[114][5]}} & 8'hb9)^
({{8{data_in[114][6]}} & 8'h59)^
({{8{data_in[114][7]}} & 8'hb2)^
({{8{data_in[115][0]}} & 8'h97)^
({{8{data_in[115][1]}} & 8'h5})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[115][2]}} & 8'ha)^
({{8{data_in[115][3]}} & 8'h14)^
({{8{data_in[115][4]}} & 8'h28)^
({{8{data_in[115][5]}} & 8'h50)^
({{8{data_in[115][6]}} & 8'ha0)^
({{8{data_in[115][7]}} & 8'h6b)^
({{8{data_in[116][0]}} & 8'hc2)^
({{8{data_in[116][1]}} & 8'haf)^
({{8{data_in[116][2]}} & 8'h75)^
({{8{data_in[116][3]}} & 8'hea)^
({{8{data_in[116][4]}} & 8'hff)^
({{8{data_in[116][5]}} & 8'hd5)^
({{8{data_in[116][6]}} & 8'h81)^
({{8{data_in[116][7]}} & 8'h29)^
({{8{data_in[117][0]}} & 8'hfd)^
({{8{data_in[117][1]}} & 8'hd1)^
({{8{data_in[117][2]}} & 8'h89)^
({{8{data_in[117][3]}} & 8'h39)^
({{8{data_in[117][4]}} & 8'h72)^
({{8{data_in[117][5]}} & 8'he4)^
({{8{data_in[117][6]}} & 8'he3)^
({{8{data_in[117][7]}} & 8'hed)^
({{8{data_in[118][0]}} & 8'h34)^
({{8{data_in[118][1]}} & 8'h68)^
({{8{data_in[118][2]}} & 8'hd0)^
({{8{data_in[118][3]}} & 8'h8b)^
({{8{data_in[118][4]}} & 8'h3d)^
({{8{data_in[118][5]}} & 8'h7a)^
({{8{data_in[118][6]}} & 8'hf4)^
({{8{data_in[118][7]}} & 8'hc3)^
({{8{data_in[119][0]}} & 8'h5b)^
({{8{data_in[119][1]}} & 8'hb6)^
({{8{data_in[119][2]}} & 8'h47)^
({{8{data_in[119][3]}} & 8'h8e)^
({{8{data_in[119][4]}} & 8'h37)^
({{8{data_in[119][5]}} & 8'h6e)^
({{8{data_in[119][6]}} & 8'hdc)^
({{8{data_in[119][7]}} & 8'h93)^
({{8{data_in[120][0]}} & 8'hd8)^
({{8{data_in[120][1]}} & 8'h9b)^
({{8{data_in[120][2]}} & 8'h1d)^
({{8{data_in[120][3]}} & 8'h3a)^
({{8{data_in[120][4]}} & 8'h74)^
({{8{data_in[120][5]}} & 8'he8)^
({{8{data_in[120][6]}} & 8'hfb)^
({{8{data_in[120][7]}} & 8'hdd)^
({{8{data_in[121][0]}} & 8'hb9)^
({{8{data_in[121][1]}} & 8'h59)^
({{8{data_in[121][2]}} & 8'hb2)^
({{8{data_in[121][3]}} & 8'h4f)^
({{8{data_in[121][4]}} & 8'h9e)^
({{8{data_in[121][5]}} & 8'h17)^
({{8{data_in[121][6]}} & 8'h2e)^
({{8{data_in[121][7]}} & 8'h5c)^
({{8{data_in[122][0]}} & 8'hb6)^
({{8{data_in[122][1]}} & 8'h47)^
({{8{data_in[122][2]}} & 8'h8e)^
({{8{data_in[122][3]}} & 8'h37)^
({{8{data_in[122][4]}} & 8'h6e)^
({{8{data_in[122][5]}} & 8'hdc)^
({{8{data_in[122][6]}} & 8'h93)^
({{8{data_in[122][7]}} & 8'hd)^
({{8{data_in[123][0]}} & 8'hc)^
({{8{data_in[123][1]}} & 8'h18)^
({{8{data_in[123][2]}} & 8'h30)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[123][3]}} & 8'h60)^
({{8{data_in[123][4]}} & 8'hc0)^
({{8{data_in[123][5]}} & 8'hab)^
({{8{data_in[123][6]}} & 8'h7d)^
({{8{data_in[123][7]}} & 8'hfa)^
({{8{data_in[124][0]}} & 8'h9)^
({{8{data_in[124][1]}} & 8'h12)^
({{8{data_in[124][2]}} & 8'h24)^
({{8{data_in[124][3]}} & 8'h48)^
({{8{data_in[124][4]}} & 8'h90)^
({{8{data_in[124][5]}} & 8'hb)^
({{8{data_in[124][6]}} & 8'h16)^
({{8{data_in[124][7]}} & 8'h2c)^
({{8{data_in[125][0]}} & 8'h4)^
({{8{data_in[125][1]}} & 8'h8)^
({{8{data_in[125][2]}} & 8'h10)^
({{8{data_in[125][3]}} & 8'h20)^
({{8{data_in[125][4]}} & 8'h40)^
({{8{data_in[125][5]}} & 8'h80)^
({{8{data_in[125][6]}} & 8'h2b)^
({{8{data_in[125][7]}} & 8'h56)^
({{8{data_in[126][0]}} & 8'hae)^
({{8{data_in[126][1]}} & 8'h77)^
({{8{data_in[126][2]}} & 8'hee)^
({{8{data_in[126][3]}} & 8'hf7)^
({{8{data_in[126][4]}} & 8'hc5)^
({{8{data_in[126][5]}} & 8'ha1)^
({{8{data_in[126][6]}} & 8'h69)^
({{8{data_in[126][7]}} & 8'hd2)^
({{8{data_in[127][0]}} & 8'h68)^
({{8{data_in[127][1]}} & 8'hd0)^
({{8{data_in[127][2]}} & 8'h8b)^
({{8{data_in[127][3]}} & 8'h3d)^
({{8{data_in[127][4]}} & 8'h7a)^
({{8{data_in[127][5]}} & 8'hf4)^
({{8{data_in[127][6]}} & 8'hc3)^
({{8{data_in[127][7]}} & 8'had)^
({{8{data_in[128][0]}} & 8'h98)^
({{8{data_in[128][1]}} & 8'h1b)^
({{8{data_in[128][2]}} & 8'h36)^
({{8{data_in[128][3]}} & 8'h6c)^
({{8{data_in[128][4]}} & 8'hd8)^
({{8{data_in[128][5]}} & 8'h9b)^
({{8{data_in[128][6]}} & 8'h1d)^
({{8{data_in[128][7]}} & 8'h3a)^
({{8{data_in[129][0]}} & 8'haf)^
({{8{data_in[129][1]}} & 8'h75)^
({{8{data_in[129][2]}} & 8'hea)^
({{8{data_in[129][3]}} & 8'hff)^
({{8{data_in[129][4]}} & 8'hd5)^
({{8{data_in[129][5]}} & 8'h81)^
({{8{data_in[129][6]}} & 8'h29)^
({{8{data_in[129][7]}} & 8'h52)^
({{8{data_in[130][0]}} & 8'h11)^
({{8{data_in[130][1]}} & 8'h22)^
({{8{data_in[130][2]}} & 8'h44)^
({{8{data_in[130][3]}} & 8'h88)^
({{8{data_in[130][4]}} & 8'h3b)^
({{8{data_in[130][5]}} & 8'h76)^
({{8{data_in[130][6]}} & 8'hec)^
({{8{data_in[130][7]}} & 8'hf3)^
({{8{data_in[131][0]}} & 8'h34)^
({{8{data_in[131][1]}} & 8'h68)^
({{8{data_in[131][2]}} & 8'hd0)^
({{8{data_in[131][3]}} & 8'h8b)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[131][4]}} & 8'h3d)^
({{8{data_in[131][5]}} & 8'h7a)^
({{8{data_in[131][6]}} & 8'hf4)^
({{8{data_in[131][7]}} & 8'hc3)^
({{8{data_in[132][0]}} & 8'hcb)^
({{8{data_in[132][1]}} & 8'hbd)^
({{8{data_in[132][2]}} & 8'h51)^
({{8{data_in[132][3]}} & 8'ha2)^
({{8{data_in[132][4]}} & 8'h6f)^
({{8{data_in[132][5]}} & 8'hde)^
({{8{data_in[132][6]}} & 8'h97)^
({{8{data_in[132][7]}} & 8'h5)^
({{8{data_in[133][0]}} & 8'h72)^
({{8{data_in[133][1]}} & 8'he4)^
({{8{data_in[133][2]}} & 8'he3)^
({{8{data_in[133][3]}} & 8'hed)^
({{8{data_in[133][4]}} & 8'hf1)^
({{8{data_in[133][5]}} & 8'hc9)^
({{8{data_in[133][6]}} & 8'hb9)^
({{8{data_in[133][7]}} & 8'h59)^
({{8{data_in[134][0]}} & 8'h55)^
({{8{data_in[134][1]}} & 8'haa)^
({{8{data_in[134][2]}} & 8'h7f)^
({{8{data_in[134][3]}} & 8'hfe)^
({{8{data_in[134][4]}} & 8'hd7)^
({{8{data_in[134][5]}} & 8'h85)^
({{8{data_in[134][6]}} & 8'h21)^
({{8{data_in[134][7]}} & 8'h42)^
({{8{data_in[135][0]}} & 8'hdf)^
({{8{data_in[135][1]}} & 8'h95)^
({{8{data_in[135][2]}} & 8'h1)^
({{8{data_in[135][3]}} & 8'h2)^
({{8{data_in[135][4]}} & 8'h4)^
({{8{data_in[135][5]}} & 8'h8)^
({{8{data_in[135][6]}} & 8'h10)^
({{8{data_in[135][7]}} & 8'h20)^
({{8{data_in[136][0]}} & 8'h5c)^
({{8{data_in[136][1]}} & 8'hb8)^
({{8{data_in[136][2]}} & 8'h5b)^
({{8{data_in[136][3]}} & 8'hb6)^
({{8{data_in[136][4]}} & 8'h47)^
({{8{data_in[136][5]}} & 8'h8e)^
({{8{data_in[136][6]}} & 8'h37)^
({{8{data_in[136][7]}} & 8'h6e)^
({{8{data_in[137][0]}} & 8'h2)^
({{8{data_in[137][1]}} & 8'h4)^
({{8{data_in[137][2]}} & 8'h8)^
({{8{data_in[137][3]}} & 8'h10)^
({{8{data_in[137][4]}} & 8'h20)^
({{8{data_in[137][5]}} & 8'h40)^
({{8{data_in[137][6]}} & 8'h80)^
({{8{data_in[137][7]}} & 8'h2b)^
({{8{data_in[138][0]}} & 8'h55)^
({{8{data_in[138][1]}} & 8'haa)^
({{8{data_in[138][2]}} & 8'h7f)^
({{8{data_in[138][3]}} & 8'hfe)^
({{8{data_in[138][4]}} & 8'hd7)^
({{8{data_in[138][5]}} & 8'h85)^
({{8{data_in[138][6]}} & 8'h21)^
({{8{data_in[138][7]}} & 8'h42)^
({{8{data_in[139][0]}} & 8'h4a)^
({{8{data_in[139][1]}} & 8'h94)^
({{8{data_in[139][2]}} & 8'h3)^
({{8{data_in[139][3]}} & 8'h6)^
({{8{data_in[139][4]}} & 8'hc)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[139][5]}} & 8'h18)^
({{8{data_in[139][6]}} & 8'h30)^
({{8{data_in[139][7]}} & 8'h60)^
({{8{data_in[140][0]}} & 8'hb9)^
({{8{data_in[140][1]}} & 8'h59)^
({{8{data_in[140][2]}} & 8'hb2)^
({{8{data_in[140][3]}} & 8'h4f)^
({{8{data_in[140][4]}} & 8'h9e)^
({{8{data_in[140][5]}} & 8'h17)^
({{8{data_in[140][6]}} & 8'h2e)^
({{8{data_in[140][7]}} & 8'h5c)^
({{8{data_in[141][0]}} & 8'h55)^
({{8{data_in[141][1]}} & 8'haa)^
({{8{data_in[141][2]}} & 8'h7f)^
({{8{data_in[141][3]}} & 8'hfe)^
({{8{data_in[141][4]}} & 8'hd7)^
({{8{data_in[141][5]}} & 8'h85)^
({{8{data_in[141][6]}} & 8'h21)^
({{8{data_in[141][7]}} & 8'h42)^
({{8{data_in[142][0]}} & 8'h42)^
({{8{data_in[142][1]}} & 8'h84)^
({{8{data_in[142][2]}} & 8'h23)^
({{8{data_in[142][3]}} & 8'h46)^
({{8{data_in[142][4]}} & 8'h8c)^
({{8{data_in[142][5]}} & 8'h33)^
({{8{data_in[142][6]}} & 8'h66)^
({{8{data_in[142][7]}} & 8'hcc)^
({{8{data_in[143][0]}} & 8'h8e)^
({{8{data_in[143][1]}} & 8'h37)^
({{8{data_in[143][2]}} & 8'h6e)^
({{8{data_in[143][3]}} & 8'hdc)^
({{8{data_in[143][4]}} & 8'h93)^
({{8{data_in[143][5]}} & 8'hd)^
({{8{data_in[143][6]}} & 8'h1a)^
({{8{data_in[143][7]}} & 8'h34)^
({{8{data_in[144][0]}} & 8'h96)^
({{8{data_in[144][1]}} & 8'h7)^
({{8{data_in[144][2]}} & 8'he)^
({{8{data_in[144][3]}} & 8'h1c)^
({{8{data_in[144][4]}} & 8'h38)^
({{8{data_in[144][5]}} & 8'h70)^
({{8{data_in[144][6]}} & 8'he0)^
({{8{data_in[144][7]}} & 8'heb)^
({{8{data_in[145][0]}} & 8'hb1)^
({{8{data_in[145][1]}} & 8'h49)^
({{8{data_in[145][2]}} & 8'h92)^
({{8{data_in[145][3]}} & 8'hf)^
({{8{data_in[145][4]}} & 8'h1e)^
({{8{data_in[145][5]}} & 8'h3c)^
({{8{data_in[145][6]}} & 8'h78)^
({{8{data_in[145][7]}} & 8'hf0)^
({{8{data_in[146][0]}} & 8'h2c)^
({{8{data_in[146][1]}} & 8'h58)^
({{8{data_in[146][2]}} & 8'hb0)^
({{8{data_in[146][3]}} & 8'h4b)^
({{8{data_in[146][4]}} & 8'h96)^
({{8{data_in[146][5]}} & 8'h7)^
({{8{data_in[146][6]}} & 8'he)^
({{8{data_in[146][7]}} & 8'h1c)^
({{8{data_in[147][0]}} & 8'h8c)^
({{8{data_in[147][1]}} & 8'h33)^
({{8{data_in[147][2]}} & 8'h66)^
({{8{data_in[147][3]}} & 8'hcc)^
({{8{data_in[147][4]}} & 8'hb3)^
({{8{data_in[147][5]}} & 8'h4d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[147][6]}} & 8'h9a)^
({{8{data_in[147][7]}} & 8'h1f)^
({{8{data_in[148][0]}} & 8'h7d)^
({{8{data_in[148][1]}} & 8'hfa)^
({{8{data_in[148][2]}} & 8'hdf)^
({{8{data_in[148][3]}} & 8'h95)^
({{8{data_in[148][4]}} & 8'h1)^
({{8{data_in[148][5]}} & 8'h2)^
({{8{data_in[148][6]}} & 8'h4)^
({{8{data_in[148][7]}} & 8'h8)^
({{8{data_in[149][0]}} & 8'hb1)^
({{8{data_in[149][1]}} & 8'h49)^
({{8{data_in[149][2]}} & 8'h92)^
({{8{data_in[149][3]}} & 8'hf)^
({{8{data_in[149][4]}} & 8'h1e)^
({{8{data_in[149][5]}} & 8'h3c)^
({{8{data_in[149][6]}} & 8'h78)^
({{8{data_in[149][7]}} & 8'hf0)^
({{8{data_in[150][0]}} & 8'hd6)^
({{8{data_in[150][1]}} & 8'h87)^
({{8{data_in[150][2]}} & 8'h25)^
({{8{data_in[150][3]}} & 8'h4a)^
({{8{data_in[150][4]}} & 8'h94)^
({{8{data_in[150][5]}} & 8'h3)^
({{8{data_in[150][6]}} & 8'h6)^
({{8{data_in[150][7]}} & 8'hc)^
({{8{data_in[151][0]}} & 8'h76)^
({{8{data_in[151][1]}} & 8'hec)^
({{8{data_in[151][2]}} & 8'hf3)^
({{8{data_in[151][3]}} & 8'hcd)^
({{8{data_in[151][4]}} & 8'hb1)^
({{8{data_in[151][5]}} & 8'h49)^
({{8{data_in[151][6]}} & 8'h92)^
({{8{data_in[151][7]}} & 8'hf)^
({{8{data_in[152][0]}} & 8'h93)^
({{8{data_in[152][1]}} & 8'hd)^
({{8{data_in[152][2]}} & 8'h1a)^
({{8{data_in[152][3]}} & 8'h34)^
({{8{data_in[152][4]}} & 8'h68)^
({{8{data_in[152][5]}} & 8'hd0)^
({{8{data_in[152][6]}} & 8'h8b)^
({{8{data_in[152][7]}} & 8'h3d)^
({{8{data_in[153][0]}} & 8'hde)^
({{8{data_in[153][1]}} & 8'h97)^
({{8{data_in[153][2]}} & 8'h5)^
({{8{data_in[153][3]}} & 8'ha)^
({{8{data_in[153][4]}} & 8'h14)^
({{8{data_in[153][5]}} & 8'h28)^
({{8{data_in[153][6]}} & 8'h50)^
({{8{data_in[153][7]}} & 8'ha0)^
({{8{data_in[154][0]}} & 8'h38)^
({{8{data_in[154][1]}} & 8'h70)^
({{8{data_in[154][2]}} & 8'he0)^
({{8{data_in[154][3]}} & 8'heb)^
({{8{data_in[154][4]}} & 8'hfd)^
({{8{data_in[154][5]}} & 8'hd1)^
({{8{data_in[154][6]}} & 8'h89)^
({{8{data_in[154][7]}} & 8'h39)^
({{8{data_in[155][0]}} & 8'hb8)^
({{8{data_in[155][1]}} & 8'h5b)^
({{8{data_in[155][2]}} & 8'hb6)^
({{8{data_in[155][3]}} & 8'h47)^
({{8{data_in[155][4]}} & 8'h8e)^
({{8{data_in[155][5]}} & 8'h37)^
({{8{data_in[155][6]}} & 8'h6e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[155][7]}} & 8'hdc)^
({{8{data_in[156][0]}} & 8'h6d)^
({{8{data_in[156][1]}} & 8'hda)^
({{8{data_in[156][2]}} & 8'h9f)^
({{8{data_in[156][3]}} & 8'h15)^
({{8{data_in[156][4]}} & 8'h2a)^
({{8{data_in[156][5]}} & 8'h54)^
({{8{data_in[156][6]}} & 8'ha8)^
({{8{data_in[156][7]}} & 8'h7b)^
({{8{data_in[157][0]}} & 8'hf8)^
({{8{data_in[157][1]}} & 8'hdb)^
({{8{data_in[157][2]}} & 8'h9d)^
({{8{data_in[157][3]}} & 8'h11)^
({{8{data_in[157][4]}} & 8'h22)^
({{8{data_in[157][5]}} & 8'h44)^
({{8{data_in[157][6]}} & 8'h88)^
({{8{data_in[157][7]}} & 8'h3b)^
({{8{data_in[158][0]}} & 8'h74)^
({{8{data_in[158][1]}} & 8'he8)^
({{8{data_in[158][2]}} & 8'hfb)^
({{8{data_in[158][3]}} & 8'hdd)^
({{8{data_in[158][4]}} & 8'h91)^
({{8{data_in[158][5]}} & 8'h9)^
({{8{data_in[158][6]}} & 8'h12)^
({{8{data_in[158][7]}} & 8'h24)^
({{8{data_in[159][0]}} & 8'hd3)^
({{8{data_in[159][1]}} & 8'h8d)^
({{8{data_in[159][2]}} & 8'h31)^
({{8{data_in[159][3]}} & 8'h62)^
({{8{data_in[159][4]}} & 8'hc4)^
({{8{data_in[159][5]}} & 8'ha3)^
({{8{data_in[159][6]}} & 8'h6d)^
({{8{data_in[159][7]}} & 8'hda)^
({{8{data_in[160][0]}} & 8'hef)^
({{8{data_in[160][1]}} & 8'hf5)^
({{8{data_in[160][2]}} & 8'hc1)^
({{8{data_in[160][3]}} & 8'ha9)^
({{8{data_in[160][4]}} & 8'h79)^
({{8{data_in[160][5]}} & 8'hf2)^
({{8{data_in[160][6]}} & 8'hcf)^
({{8{data_in[160][7]}} & 8'hb5)^
({{8{data_in[161][0]}} & 8'hc0)^
({{8{data_in[161][1]}} & 8'hab)^
({{8{data_in[161][2]}} & 8'h7d)^
({{8{data_in[161][3]}} & 8'hfa)^
({{8{data_in[161][4]}} & 8'hdf)^
({{8{data_in[161][5]}} & 8'h95)^
({{8{data_in[161][6]}} & 8'h1)^
({{8{data_in[161][7]}} & 8'h2)^
({{8{data_in[162][0]}} & 8'hd2)^
({{8{data_in[162][1]}} & 8'h8f)^
({{8{data_in[162][2]}} & 8'h35)^
({{8{data_in[162][3]}} & 8'h6a)^
({{8{data_in[162][4]}} & 8'hd4)^
({{8{data_in[162][5]}} & 8'h83)^
({{8{data_in[162][6]}} & 8'h2d)^
({{8{data_in[162][7]}} & 8'h5a)^
({{8{data_in[163][0]}} & 8'h23)^
({{8{data_in[163][1]}} & 8'h46)^
({{8{data_in[163][2]}} & 8'h8c)^
({{8{data_in[163][3]}} & 8'h33)^
({{8{data_in[163][4]}} & 8'h66)^
({{8{data_in[163][5]}} & 8'hcc)^
({{8{data_in[163][6]}} & 8'hb3)^
({{8{data_in[163][7]}} & 8'h4d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[164][0]}} & 8'h1a)^
({{8{data_in[164][1]}} & 8'h34)^
({{8{data_in[164][2]}} & 8'h68)^
({{8{data_in[164][3]}} & 8'hd0)^
({{8{data_in[164][4]}} & 8'h8b)^
({{8{data_in[164][5]}} & 8'h3d)^
({{8{data_in[164][6]}} & 8'h7a)^
({{8{data_in[164][7]}} & 8'hf4)^
({{8{data_in[165][0]}} & 8'h2a)^
({{8{data_in[165][1]}} & 8'h54)^
({{8{data_in[165][2]}} & 8'ha8)^
({{8{data_in[165][3]}} & 8'h7b)^
({{8{data_in[165][4]}} & 8'hf6)^
({{8{data_in[165][5]}} & 8'hc7)^
({{8{data_in[165][6]}} & 8'ha5)^
({{8{data_in[165][7]}} & 8'h61)^
({{8{data_in[166][0]}} & 8'h8b)^
({{8{data_in[166][1]}} & 8'h3d)^
({{8{data_in[166][2]}} & 8'h7a)^
({{8{data_in[166][3]}} & 8'hf4)^
({{8{data_in[166][4]}} & 8'hc3)^
({{8{data_in[166][5]}} & 8'had)^
({{8{data_in[166][6]}} & 8'h71)^
({{8{data_in[166][7]}} & 8'he2)^
({{8{data_in[167][0]}} & 8'h7)^
({{8{data_in[167][1]}} & 8'he)^
({{8{data_in[167][2]}} & 8'h1c)^
({{8{data_in[167][3]}} & 8'h38)^
({{8{data_in[167][4]}} & 8'h70)^
({{8{data_in[167][5]}} & 8'he0)^
({{8{data_in[167][6]}} & 8'heb)^
({{8{data_in[167][7]}} & 8'hfd)^
({{8{data_in[168][0]}} & 8'h6c)^
({{8{data_in[168][1]}} & 8'hd8)^
({{8{data_in[168][2]}} & 8'h9b)^
({{8{data_in[168][3]}} & 8'h1d)^
({{8{data_in[168][4]}} & 8'h3a)^
({{8{data_in[168][5]}} & 8'h74)^
({{8{data_in[168][6]}} & 8'he8)^
({{8{data_in[168][7]}} & 8'hfb)^
({{8{data_in[169][0]}} & 8'hb1)^
({{8{data_in[169][1]}} & 8'h49)^
({{8{data_in[169][2]}} & 8'h92)^
({{8{data_in[169][3]}} & 8'hf)^
({{8{data_in[169][4]}} & 8'h1e)^
({{8{data_in[169][5]}} & 8'h3c)^
({{8{data_in[169][6]}} & 8'h78)^
({{8{data_in[169][7]}} & 8'hf0)^
({{8{data_in[170][0]}} & 8'h5f)^
({{8{data_in[170][1]}} & 8'hbe)^
({{8{data_in[170][2]}} & 8'h57)^
({{8{data_in[170][3]}} & 8'hae)^
({{8{data_in[170][4]}} & 8'h77)^
({{8{data_in[170][5]}} & 8'hee)^
({{8{data_in[170][6]}} & 8'hf7)^
({{8{data_in[170][7]}} & 8'hc5)^
({{8{data_in[171][0]}} & 8'hb4)^
({{8{data_in[171][1]}} & 8'h43)^
({{8{data_in[171][2]}} & 8'h86)^
({{8{data_in[171][3]}} & 8'h27)^
({{8{data_in[171][4]}} & 8'h4e)^
({{8{data_in[171][5]}} & 8'h9c)^
({{8{data_in[171][6]}} & 8'h13)^
({{8{data_in[171][7]}} & 8'h26)^
({{8{data_in[172][0]}} & 8'h6d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[172][1]}} & 8'hda)^
({{8{data_in[172][2]}} & 8'h9f)^
({{8{data_in[172][3]}} & 8'h15)^
({{8{data_in[172][4]}} & 8'h2a)^
({{8{data_in[172][5]}} & 8'h54)^
({{8{data_in[172][6]}} & 8'ha8)^
({{8{data_in[172][7]}} & 8'h7b)^
({{8{data_in[173][0]}} & 8'hfb)^
({{8{data_in[173][1]}} & 8'hdd)^
({{8{data_in[173][2]}} & 8'h91)^
({{8{data_in[173][3]}} & 8'h9)^
({{8{data_in[173][4]}} & 8'h12)^
({{8{data_in[173][5]}} & 8'h24)^
({{8{data_in[173][6]}} & 8'h48)^
({{8{data_in[173][7]}} & 8'h90)^
({{8{data_in[174][0]}} & 8'h4)^
({{8{data_in[174][1]}} & 8'h8)^
({{8{data_in[174][2]}} & 8'h10)^
({{8{data_in[174][3]}} & 8'h20)^
({{8{data_in[174][4]}} & 8'h40)^
({{8{data_in[174][5]}} & 8'h80)^
({{8{data_in[174][6]}} & 8'h2b)^
({{8{data_in[174][7]}} & 8'h56)^
({{8{data_in[175][0]}} & 8'h53)^
({{8{data_in[175][1]}} & 8'ha6)^
({{8{data_in[175][2]}} & 8'h67)^
({{8{data_in[175][3]}} & 8'hce)^
({{8{data_in[175][4]}} & 8'hb7)^
({{8{data_in[175][5]}} & 8'h45)^
({{8{data_in[175][6]}} & 8'h8a)^
({{8{data_in[175][7]}} & 8'h3f)^
({{8{data_in[176][0]}} & 8'h9e)^
({{8{data_in[176][1]}} & 8'h17)^
({{8{data_in[176][2]}} & 8'h2e)^
({{8{data_in[176][3]}} & 8'h5c)^
({{8{data_in[176][4]}} & 8'hb8)^
({{8{data_in[176][5]}} & 8'h5b)^
({{8{data_in[176][6]}} & 8'hb6)^
({{8{data_in[176][7]}} & 8'h47)^
({{8{data_in[177][0]}} & 8'h9f)^
({{8{data_in[177][1]}} & 8'h15)^
({{8{data_in[177][2]}} & 8'h2a)^
({{8{data_in[177][3]}} & 8'h54)^
({{8{data_in[177][4]}} & 8'ha8)^
({{8{data_in[177][5]}} & 8'h7b)^
({{8{data_in[177][6]}} & 8'hf6)^
({{8{data_in[177][7]}} & 8'hc7)^
({{8{data_in[178][0]}} & 8'h8e)^
({{8{data_in[178][1]}} & 8'h37)^
({{8{data_in[178][2]}} & 8'h6e)^
({{8{data_in[178][3]}} & 8'hdc)^
({{8{data_in[178][4]}} & 8'h93)^
({{8{data_in[178][5]}} & 8'hd)^
({{8{data_in[178][6]}} & 8'h1a)^
({{8{data_in[178][7]}} & 8'h34)^
({{8{data_in[179][0]}} & 8'hf9)^
({{8{data_in[179][1]}} & 8'hd9)^
({{8{data_in[179][2]}} & 8'h99)^
({{8{data_in[179][3]}} & 8'h19)^
({{8{data_in[179][4]}} & 8'h32)^
({{8{data_in[179][5]}} & 8'h64)^
({{8{data_in[179][6]}} & 8'hc8)^
({{8{data_in[179][7]}} & 8'hbb)^
({{8{data_in[180][0]}} & 8'h38)^
({{8{data_in[180][1]}} & 8'h70)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[180][2]}} & 8'he0)^
({{8{data_in[180][3]}} & 8'heb)^
({{8{data_in[180][4]}} & 8'hfd)^
({{8{data_in[180][5]}} & 8'hd1)^
({{8{data_in[180][6]}} & 8'h89)^
({{8{data_in[180][7]}} & 8'h39)^
({{8{data_in[181][0]}} & 8'hf7)^
({{8{data_in[181][1]}} & 8'hc5)^
({{8{data_in[181][2]}} & 8'ha1)^
({{8{data_in[181][3]}} & 8'h69)^
({{8{data_in[181][4]}} & 8'hd2)^
({{8{data_in[181][5]}} & 8'h8f)^
({{8{data_in[181][6]}} & 8'h35)^
({{8{data_in[181][7]}} & 8'h6a)^
({{8{data_in[182][0]}} & 8'hc7)^
({{8{data_in[182][1]}} & 8'ha5)^
({{8{data_in[182][2]}} & 8'h61)^
({{8{data_in[182][3]}} & 8'hc2)^
({{8{data_in[182][4]}} & 8'haf)^
({{8{data_in[182][5]}} & 8'h75)^
({{8{data_in[182][6]}} & 8'hea)^
({{8{data_in[182][7]}} & 8'hff)^
({{8{data_in[183][0]}} & 8'he8)^
({{8{data_in[183][1]}} & 8'hfb)^
({{8{data_in[183][2]}} & 8'hdd)^
({{8{data_in[183][3]}} & 8'h91)^
({{8{data_in[183][4]}} & 8'h9)^
({{8{data_in[183][5]}} & 8'h12)^
({{8{data_in[183][6]}} & 8'h24)^
({{8{data_in[183][7]}} & 8'h48)^
({{8{data_in[184][0]}} & 8'h53)^
({{8{data_in[184][1]}} & 8'ha6)^
({{8{data_in[184][2]}} & 8'h67)^
({{8{data_in[184][3]}} & 8'hce)^
({{8{data_in[184][4]}} & 8'hb7)^
({{8{data_in[184][5]}} & 8'h45)^
({{8{data_in[184][6]}} & 8'h8a)^
({{8{data_in[184][7]}} & 8'h3f)^
({{8{data_in[185][0]}} & 8'he8)^
({{8{data_in[185][1]}} & 8'hfb)^
({{8{data_in[185][2]}} & 8'hdd)^
({{8{data_in[185][3]}} & 8'h91)^
({{8{data_in[185][4]}} & 8'h9)^
({{8{data_in[185][5]}} & 8'h12)^
({{8{data_in[185][6]}} & 8'h24)^
({{8{data_in[185][7]}} & 8'h48)^
({{8{data_in[186][0]}} & 8'h17)^
({{8{data_in[186][1]}} & 8'h2e)^
({{8{data_in[186][2]}} & 8'h5c)^
({{8{data_in[186][3]}} & 8'hb8)^
({{8{data_in[186][4]}} & 8'h5b)^
({{8{data_in[186][5]}} & 8'hb6)^
({{8{data_in[186][6]}} & 8'h47)^
({{8{data_in[186][7]}} & 8'h8e)^
({{8{data_in[187][0]}} & 8'h3a)^
({{8{data_in[187][1]}} & 8'h74)^
({{8{data_in[187][2]}} & 8'he8)^
({{8{data_in[187][3]}} & 8'hfb)^
({{8{data_in[187][4]}} & 8'hdd)^
({{8{data_in[187][5]}} & 8'h91)^
({{8{data_in[187][6]}} & 8'h9)^
({{8{data_in[187][7]}} & 8'h12)^
({{8{data_in[188][0]}} & 8'h3b)^
({{8{data_in[188][1]}} & 8'h76)^
({{8{data_in[188][2]}} & 8'hec)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[188][3]}} & 8'hf3)^
({{8{data_in[188][4]}} & 8'hcd)^
({{8{data_in[188][5]}} & 8'hb1)^
({{8{data_in[188][6]}} & 8'h49)^
({{8{data_in[188][7]}} & 8'h92)^
({{8{data_in[189][0]}} & 8'h46)^
({{8{data_in[189][1]}} & 8'h8c)^
({{8{data_in[189][2]}} & 8'h33)^
({{8{data_in[189][3]}} & 8'h66)^
({{8{data_in[189][4]}} & 8'hcc)^
({{8{data_in[189][5]}} & 8'hb3)^
({{8{data_in[189][6]}} & 8'h4d)^
({{8{data_in[189][7]}} & 8'h9a)^
({{8{data_in[190][0]}} & 8'h95)^
({{8{data_in[190][1]}} & 8'h1)^
({{8{data_in[190][2]}} & 8'h2)^
({{8{data_in[190][3]}} & 8'h4)^
({{8{data_in[190][4]}} & 8'h8)^
({{8{data_in[190][5]}} & 8'h10)^
({{8{data_in[190][6]}} & 8'h20)^
({{8{data_in[190][7]}} & 8'h40)^
({{8{data_in[191][0]}} & 8'h66)^
({{8{data_in[191][1]}} & 8'hcc)^
({{8{data_in[191][2]}} & 8'hb3)^
({{8{data_in[191][3]}} & 8'h4d)^
({{8{data_in[191][4]}} & 8'h9a)^
({{8{data_in[191][5]}} & 8'h1f)^
({{8{data_in[191][6]}} & 8'h3e)^
({{8{data_in[191][7]}} & 8'h7c)^
({{8{data_in[192][0]}} & 8'hbb)^
({{8{data_in[192][1]}} & 8'h5d)^
({{8{data_in[192][2]}} & 8'hba)^
({{8{data_in[192][3]}} & 8'h5f)^
({{8{data_in[192][4]}} & 8'hbe)^
({{8{data_in[192][5]}} & 8'h57)^
({{8{data_in[192][6]}} & 8'hae)^
({{8{data_in[192][7]}} & 8'h77)^
({{8{data_in[193][0]}} & 8'h3e)^
({{8{data_in[193][1]}} & 8'h7c)^
({{8{data_in[193][2]}} & 8'hf8)^
({{8{data_in[193][3]}} & 8'hdb)^
({{8{data_in[193][4]}} & 8'h9d)^
({{8{data_in[193][5]}} & 8'h11)^
({{8{data_in[193][6]}} & 8'h22)^
({{8{data_in[193][7]}} & 8'h44)^
({{8{data_in[194][0]}} & 8'h40)^
({{8{data_in[194][1]}} & 8'h80)^
({{8{data_in[194][2]}} & 8'h2b)^
({{8{data_in[194][3]}} & 8'h56)^
({{8{data_in[194][4]}} & 8'hac)^
({{8{data_in[194][5]}} & 8'h73)^
({{8{data_in[194][6]}} & 8'he6)^
({{8{data_in[194][7]}} & 8'he7)^
({{8{data_in[195][0]}} & 8'h6b)^
({{8{data_in[195][1]}} & 8'hd6)^
({{8{data_in[195][2]}} & 8'h87)^
({{8{data_in[195][3]}} & 8'h25)^
({{8{data_in[195][4]}} & 8'h4a)^
({{8{data_in[195][5]}} & 8'h94)^
({{8{data_in[195][6]}} & 8'h3)^
({{8{data_in[195][7]}} & 8'h6)^
({{8{data_in[196][0]}} & 8'h4d)^
({{8{data_in[196][1]}} & 8'h9a)^
({{8{data_in[196][2]}} & 8'h1f)^
({{8{data_in[196][3]}} & 8'h3e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[196][4]}} & 8'h7c)^
({{8{data_in[196][5]}} & 8'hf8)^
({{8{data_in[196][6]}} & 8'hdb)^
({{8{data_in[196][7]}} & 8'h9d)^
({{8{data_in[197][0]}} & 8'h9a)^
({{8{data_in[197][1]}} & 8'h1f)^
({{8{data_in[197][2]}} & 8'h3e)^
({{8{data_in[197][3]}} & 8'h7c)^
({{8{data_in[197][4]}} & 8'hf8)^
({{8{data_in[197][5]}} & 8'hdb)^
({{8{data_in[197][6]}} & 8'h9d)^
({{8{data_in[197][7]}} & 8'h11)^
({{8{data_in[198][0]}} & 8'h9c)^
({{8{data_in[198][1]}} & 8'h13)^
({{8{data_in[198][2]}} & 8'h26)^
({{8{data_in[198][3]}} & 8'h4c)^
({{8{data_in[198][4]}} & 8'h98)^
({{8{data_in[198][5]}} & 8'h1b)^
({{8{data_in[198][6]}} & 8'h36)^
({{8{data_in[198][7]}} & 8'h6c)^
({{8{data_in[199][0]}} & 8'h93)^
({{8{data_in[199][1]}} & 8'hd)^
({{8{data_in[199][2]}} & 8'h1a)^
({{8{data_in[199][3]}} & 8'h34)^
({{8{data_in[199][4]}} & 8'h68)^
({{8{data_in[199][5]}} & 8'hd0)^
({{8{data_in[199][6]}} & 8'h8b)^
({{8{data_in[199][7]}} & 8'h3d)^
({{8{data_in[200][0]}} & 8'h12)^
({{8{data_in[200][1]}} & 8'h24)^
({{8{data_in[200][2]}} & 8'h48)^
({{8{data_in[200][3]}} & 8'h90)^
({{8{data_in[200][4]}} & 8'hb)^
({{8{data_in[200][5]}} & 8'h16)^
({{8{data_in[200][6]}} & 8'h2c)^
({{8{data_in[200][7]}} & 8'h58)^
({{8{data_in[201][0]}} & 8'hff)^
({{8{data_in[201][1]}} & 8'hd5)^
({{8{data_in[201][2]}} & 8'h81)^
({{8{data_in[201][3]}} & 8'h29)^
({{8{data_in[201][4]}} & 8'h52)^
({{8{data_in[201][5]}} & 8'ha4)^
({{8{data_in[201][6]}} & 8'h63)^
({{8{data_in[201][7]}} & 8'hc6)^
({{8{data_in[202][0]}} & 8'h35)^
({{8{data_in[202][1]}} & 8'h6a)^
({{8{data_in[202][2]}} & 8'hd4)^
({{8{data_in[202][3]}} & 8'h83)^
({{8{data_in[202][4]}} & 8'h2d)^
({{8{data_in[202][5]}} & 8'h5a)^
({{8{data_in[202][6]}} & 8'hb4)^
({{8{data_in[202][7]}} & 8'h43)^
({{8{data_in[203][0]}} & 8'hcf)^
({{8{data_in[203][1]}} & 8'hb5)^
({{8{data_in[203][2]}} & 8'h41)^
({{8{data_in[203][3]}} & 8'h82)^
({{8{data_in[203][4]}} & 8'h2f)^
({{8{data_in[203][5]}} & 8'h5e)^
({{8{data_in[203][6]}} & 8'hbc)^
({{8{data_in[203][7]}} & 8'h53)^
({{8{data_in[204][0]}} & 8'h87)^
({{8{data_in[204][1]}} & 8'h25)^
({{8{data_in[204][2]}} & 8'h4a)^
({{8{data_in[204][3]}} & 8'h94)^
({{8{data_in[204][4]}} & 8'h3)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[204][5]}} & 8'h6)^
({{8{data_in[204][6]}} & 8'hc)^
({{8{data_in[204][7]}} & 8'h18)^
({{8{data_in[205][0]}} & 8'hf4)^
({{8{data_in[205][1]}} & 8'hc3)^
({{8{data_in[205][2]}} & 8'had)^
({{8{data_in[205][3]}} & 8'h71)^
({{8{data_in[205][4]}} & 8'he2)^
({{8{data_in[205][5]}} & 8'hef)^
({{8{data_in[205][6]}} & 8'hf5)^
({{8{data_in[205][7]}} & 8'hc1)^
({{8{data_in[206][0]}} & 8'hbc)^
({{8{data_in[206][1]}} & 8'h53)^
({{8{data_in[206][2]}} & 8'ha6)^
({{8{data_in[206][3]}} & 8'h67)^
({{8{data_in[206][4]}} & 8'hce)^
({{8{data_in[206][5]}} & 8'hb7)^
({{8{data_in[206][6]}} & 8'h45)^
({{8{data_in[206][7]}} & 8'h8a)^
({{8{data_in[207][0]}} & 8'h8)^
({{8{data_in[207][1]}} & 8'h10)^
({{8{data_in[207][2]}} & 8'h20)^
({{8{data_in[207][3]}} & 8'h40)^
({{8{data_in[207][4]}} & 8'h80)^
({{8{data_in[207][5]}} & 8'h2b)^
({{8{data_in[207][6]}} & 8'h56)^
({{8{data_in[207][7]}} & 8'hac)^
({{8{data_in[208][0]}} & 8'h4c)^
({{8{data_in[208][1]}} & 8'h98)^
({{8{data_in[208][2]}} & 8'h1b)^
({{8{data_in[208][3]}} & 8'h36)^
({{8{data_in[208][4]}} & 8'h6c)^
({{8{data_in[208][5]}} & 8'hd8)^
({{8{data_in[208][6]}} & 8'h9b)^
({{8{data_in[208][7]}} & 8'h1d)^
({{8{data_in[209][0]}} & 8'hc6)^
({{8{data_in[209][1]}} & 8'ha7)^
({{8{data_in[209][2]}} & 8'h65)^
({{8{data_in[209][3]}} & 8'hca)^
({{8{data_in[209][4]}} & 8'hbf)^
({{8{data_in[209][5]}} & 8'h55)^
({{8{data_in[209][6]}} & 8'haa)^
({{8{data_in[209][7]}} & 8'h7f)^
({{8{data_in[210][0]}} & 8'hd3)^
({{8{data_in[210][1]}} & 8'h8d)^
({{8{data_in[210][2]}} & 8'h31)^
({{8{data_in[210][3]}} & 8'h62)^
({{8{data_in[210][4]}} & 8'hc4)^
({{8{data_in[210][5]}} & 8'ha3)^
({{8{data_in[210][6]}} & 8'h6d)^
({{8{data_in[210][7]}} & 8'hda)^
({{8{data_in[211][0]}} & 8'h50)^
({{8{data_in[211][1]}} & 8'ha0)^
({{8{data_in[211][2]}} & 8'h6b)^
({{8{data_in[211][3]}} & 8'hd6)^
({{8{data_in[211][4]}} & 8'h87)^
({{8{data_in[211][5]}} & 8'h25)^
({{8{data_in[211][6]}} & 8'h4a)^
({{8{data_in[211][7]}} & 8'h94)^
({{8{data_in[212][0]}} & 8'hb0)^
({{8{data_in[212][1]}} & 8'h4b)^
({{8{data_in[212][2]}} & 8'h96)^
({{8{data_in[212][3]}} & 8'h7)^
({{8{data_in[212][4]}} & 8'he)^
({{8{data_in[212][5]}} & 8'h1c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[212][6]}} & 8'h38)^
({{8{data_in[212][7]}} & 8'h70)^
({{8{data_in[213][0]}} & 8'hb8)^
({{8{data_in[213][1]}} & 8'h5b)^
({{8{data_in[213][2]}} & 8'hb6)^
({{8{data_in[213][3]}} & 8'h47)^
({{8{data_in[213][4]}} & 8'h8e)^
({{8{data_in[213][5]}} & 8'h37)^
({{8{data_in[213][6]}} & 8'h6e)^
({{8{data_in[213][7]}} & 8'hdc)^
({{8{data_in[214][0]}} & 8'hec)^
({{8{data_in[214][1]}} & 8'hf3)^
({{8{data_in[214][2]}} & 8'hcd)^
({{8{data_in[214][3]}} & 8'hb1)^
({{8{data_in[214][4]}} & 8'h49)^
({{8{data_in[214][5]}} & 8'h92)^
({{8{data_in[214][6]}} & 8'hf)^
({{8{data_in[214][7]}} & 8'h1e)^
({{8{data_in[215][0]}} & 8'hf9)^
({{8{data_in[215][1]}} & 8'hd9)^
({{8{data_in[215][2]}} & 8'h99)^
({{8{data_in[215][3]}} & 8'h19)^
({{8{data_in[215][4]}} & 8'h32)^
({{8{data_in[215][5]}} & 8'h64)^
({{8{data_in[215][6]}} & 8'hc8)^
({{8{data_in[215][7]}} & 8'hbb)^
({{8{data_in[216][0]}} & 8'hcb)^
({{8{data_in[216][1]}} & 8'hbd)^
({{8{data_in[216][2]}} & 8'h51)^
({{8{data_in[216][3]}} & 8'ha2)^
({{8{data_in[216][4]}} & 8'h6f)^
({{8{data_in[216][5]}} & 8'hde)^
({{8{data_in[216][6]}} & 8'h97)^
({{8{data_in[216][7]}} & 8'h5)^
({{8{data_in[217][0]}} & 8'h22)^
({{8{data_in[217][1]}} & 8'h44)^
({{8{data_in[217][2]}} & 8'h88)^
({{8{data_in[217][3]}} & 8'h3b)^
({{8{data_in[217][4]}} & 8'h76)^
({{8{data_in[217][5]}} & 8'hec)^
({{8{data_in[217][6]}} & 8'hf3)^
({{8{data_in[217][7]}} & 8'hcd)^
({{8{data_in[218][0]}} & 8'h1e)^
({{8{data_in[218][1]}} & 8'h3c)^
({{8{data_in[218][2]}} & 8'h78)^
({{8{data_in[218][3]}} & 8'hf0)^
({{8{data_in[218][4]}} & 8'hcb)^
({{8{data_in[218][5]}} & 8'hbd)^
({{8{data_in[218][6]}} & 8'h51)^
({{8{data_in[218][7]}} & 8'ha2)^
({{8{data_in[219][0]}} & 8'hc8)^
({{8{data_in[219][1]}} & 8'hbb)^
({{8{data_in[219][2]}} & 8'h5d)^
({{8{data_in[219][3]}} & 8'hba)^
({{8{data_in[219][4]}} & 8'h5f)^
({{8{data_in[219][5]}} & 8'hbe)^
({{8{data_in[219][6]}} & 8'h57)^
({{8{data_in[219][7]}} & 8'hae)^
({{8{data_in[220][0]}} & 8'hdc)^
({{8{data_in[220][1]}} & 8'h93)^
({{8{data_in[220][2]}} & 8'hd)^
({{8{data_in[220][3]}} & 8'h1a)^
({{8{data_in[220][4]}} & 8'h34)^
({{8{data_in[220][5]}} & 8'h68)^
({{8{data_in[220][6]}} & 8'hd0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[220][7]}} & 8'h8b)^
({{8{data_in[221][0]}} & 8'ha2)^
({{8{data_in[221][1]}} & 8'h6f)^
({{8{data_in[221][2]}} & 8'hde)^
({{8{data_in[221][3]}} & 8'h97)^
({{8{data_in[221][4]}} & 8'h5)^
({{8{data_in[221][5]}} & 8'ha)^
({{8{data_in[221][6]}} & 8'h14)^
({{8{data_in[221][7]}} & 8'h28)^
({{8{data_in[222][0]}} & 8'hdb)^
({{8{data_in[222][1]}} & 8'h9d)^
({{8{data_in[222][2]}} & 8'h11)^
({{8{data_in[222][3]}} & 8'h22)^
({{8{data_in[222][4]}} & 8'h44)^
({{8{data_in[222][5]}} & 8'h88)^
({{8{data_in[222][6]}} & 8'h3b)^
({{8{data_in[222][7]}} & 8'h76)^
({{8{data_in[223][0]}} & 8'hb6)^
({{8{data_in[223][1]}} & 8'h47)^
({{8{data_in[223][2]}} & 8'h8e)^
({{8{data_in[223][3]}} & 8'h37)^
({{8{data_in[223][4]}} & 8'h6e)^
({{8{data_in[223][5]}} & 8'hdc)^
({{8{data_in[223][6]}} & 8'h93)^
({{8{data_in[223][7]}} & 8'hd)^
({{8{data_in[224][0]}} & 8'h89)^
({{8{data_in[224][1]}} & 8'h39)^
({{8{data_in[224][2]}} & 8'h72)^
({{8{data_in[224][3]}} & 8'he4)^
({{8{data_in[224][4]}} & 8'he3)^
({{8{data_in[224][5]}} & 8'hd)^
({{8{data_in[224][6]}} & 8'hf1)^
({{8{data_in[224][7]}} & 8'hc9)^
({{8{data_in[225][0]}} & 8'he2)^
({{8{data_in[225][1]}} & 8'hef)^
({{8{data_in[225][2]}} & 8'hf5)^
({{8{data_in[225][3]}} & 8'hc1)^
({{8{data_in[225][4]}} & 8'ha9)^
({{8{data_in[225][5]}} & 8'h79)^
({{8{data_in[225][6]}} & 8'hf2)^
({{8{data_in[225][7]}} & 8'hcf)^
({{8{data_in[226][0]}} & 8'h9f)^
({{8{data_in[226][1]}} & 8'h15)^
({{8{data_in[226][2]}} & 8'h2a)^
({{8{data_in[226][3]}} & 8'h54)^
({{8{data_in[226][4]}} & 8'ha8)^
({{8{data_in[226][5]}} & 8'h7b)^
({{8{data_in[226][6]}} & 8'hf6)^
({{8{data_in[226][7]}} & 8'hc7)^
({{8{data_in[227][0]}} & 8'h95)^
({{8{data_in[227][1]}} & 8'h1)^
({{8{data_in[227][2]}} & 8'h2)^
({{8{data_in[227][3]}} & 8'h4)^
({{8{data_in[227][4]}} & 8'h8)^
({{8{data_in[227][5]}} & 8'h10)^
({{8{data_in[227][6]}} & 8'h20)^
({{8{data_in[227][7]}} & 8'h40)^
({{8{data_in[228][0]}} & 8'h9c)^
({{8{data_in[228][1]}} & 8'h13)^
({{8{data_in[228][2]}} & 8'h26)^
({{8{data_in[228][3]}} & 8'h4c)^
({{8{data_in[228][4]}} & 8'h98)^
({{8{data_in[228][5]}} & 8'h1b)^
({{8{data_in[228][6]}} & 8'h36)^
({{8{data_in[228][7]}} & 8'h6c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[229][0]}} & 8'h9e)^
({{8{data_in[229][1]}} & 8'h17)^
({{8{data_in[229][2]}} & 8'h2e)^
({{8{data_in[229][3]}} & 8'h5c)^
({{8{data_in[229][4]}} & 8'hb8)^
({{8{data_in[229][5]}} & 8'h5b)^
({{8{data_in[229][6]}} & 8'hb6)^
({{8{data_in[229][7]}} & 8'h47)^
({{8{data_in[230][0]}} & 8'h1a)^
({{8{data_in[230][1]}} & 8'h34)^
({{8{data_in[230][2]}} & 8'h68)^
({{8{data_in[230][3]}} & 8'hd0)^
({{8{data_in[230][4]}} & 8'h8b)^
({{8{data_in[230][5]}} & 8'h3d)^
({{8{data_in[230][6]}} & 8'h7a)^
({{8{data_in[230][7]}} & 8'hf4)^
({{8{data_in[231][0]}} & 8'h6f)^
({{8{data_in[231][1]}} & 8'hde)^
({{8{data_in[231][2]}} & 8'h97)^
({{8{data_in[231][3]}} & 8'h5)^
({{8{data_in[231][4]}} & 8'ha)^
({{8{data_in[231][5]}} & 8'h14)^
({{8{data_in[231][6]}} & 8'h28)^
({{8{data_in[231][7]}} & 8'h50)^
({{8{data_in[232][0]}} & 8'h69)^
({{8{data_in[232][1]}} & 8'hd2)^
({{8{data_in[232][2]}} & 8'h8f)^
({{8{data_in[232][3]}} & 8'h35)^
({{8{data_in[232][4]}} & 8'h6a)^
({{8{data_in[232][5]}} & 8'hd4)^
({{8{data_in[232][6]}} & 8'h83)^
({{8{data_in[232][7]}} & 8'h2d)^
({{8{data_in[233][0]}} & 8'h8f)^
({{8{data_in[233][1]}} & 8'h35)^
({{8{data_in[233][2]}} & 8'h6a)^
({{8{data_in[233][3]}} & 8'hd4)^
({{8{data_in[233][4]}} & 8'h83)^
({{8{data_in[233][5]}} & 8'h2d)^
({{8{data_in[233][6]}} & 8'h5a)^
({{8{data_in[233][7]}} & 8'hb4)^
({{8{data_in[234][0]}} & 8'h9e)^
({{8{data_in[234][1]}} & 8'h17)^
({{8{data_in[234][2]}} & 8'h2e)^
({{8{data_in[234][3]}} & 8'h5c)^
({{8{data_in[234][4]}} & 8'hb8)^
({{8{data_in[234][5]}} & 8'h5b)^
({{8{data_in[234][6]}} & 8'hb6)^
({{8{data_in[234][7]}} & 8'h47)^
({{8{data_in[235][0]}} & 8'h5f)^
({{8{data_in[235][1]}} & 8'hbe)^
({{8{data_in[235][2]}} & 8'h57)^
({{8{data_in[235][3]}} & 8'hae)^
({{8{data_in[235][4]}} & 8'h77)^
({{8{data_in[235][5]}} & 8'hee)^
({{8{data_in[235][6]}} & 8'hf7)^
({{8{data_in[235][7]}} & 8'hc5)^
({{8{data_in[236][0]}} & 8'h8c)^
({{8{data_in[236][1]}} & 8'h33)^
({{8{data_in[236][2]}} & 8'h66)^
({{8{data_in[236][3]}} & 8'hcc)^
({{8{data_in[236][4]}} & 8'hb3)^
({{8{data_in[236][5]}} & 8'h4d)^
({{8{data_in[236][6]}} & 8'h9a)^
({{8{data_in[236][7]}} & 8'h1f)^
({{8{data_in[237][0]}} & 8'h17)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[237][1]}} & 8'h2e)^
({{8{data_in[237][2]}} & 8'h5c)^
({{8{data_in[237][3]}} & 8'hb8)^
({{8{data_in[237][4]}} & 8'h5b)^
({{8{data_in[237][5]}} & 8'hb6)^
({{8{data_in[237][6]}} & 8'h47)^
({{8{data_in[237][7]}} & 8'h8e)^
({{8{data_in[238][0]}} & 8'h96)^
({{8{data_in[238][1]}} & 8'h7)^
({{8{data_in[238][2]}} & 8'he)^
({{8{data_in[238][3]}} & 8'h1c)^
({{8{data_in[238][4]}} & 8'h38)^
({{8{data_in[238][5]}} & 8'h70)^
({{8{data_in[238][6]}} & 8'he0)^
({{8{data_in[238][7]}} & 8'heb)^
({{8{data_in[239][0]}} & 8'h95)^
({{8{data_in[239][1]}} & 8'h1)^
({{8{data_in[239][2]}} & 8'h2)^
({{8{data_in[239][3]}} & 8'h4)^
({{8{data_in[239][4]}} & 8'h8)^
({{8{data_in[239][5]}} & 8'h10)^
({{8{data_in[239][6]}} & 8'h20)^
({{8{data_in[239][7]}} & 8'h40)^
({{8{data_in[240][0]}} & 8'h89)^
({{8{data_in[240][1]}} & 8'h39)^
({{8{data_in[240][2]}} & 8'h72)^
({{8{data_in[240][3]}} & 8'he4)^
({{8{data_in[240][4]}} & 8'he3)^
({{8{data_in[240][5]}} & 8'hd)^
({{8{data_in[240][6]}} & 8'hf1)^
({{8{data_in[240][7]}} & 8'hc9)^
({{8{data_in[241][0]}} & 8'hfe)^
({{8{data_in[241][1]}} & 8'hd7)^
({{8{data_in[241][2]}} & 8'h85)^
({{8{data_in[241][3]}} & 8'h21)^
({{8{data_in[241][4]}} & 8'h42)^
({{8{data_in[241][5]}} & 8'h84)^
({{8{data_in[241][6]}} & 8'h23)^
({{8{data_in[241][7]}} & 8'h46);

data_out[246] = ({8{data_in[0][0]}} & 8'he9)^
({8{data_in[0][1]}} & 8'hf9)^
({8{data_in[0][2]}} & 8'hd9)^
({8{data_in[0][3]}} & 8'h99)^
({8{data_in[0][4]}} & 8'h19)^
({8{data_in[0][5]}} & 8'h32)^
({8{data_in[0][6]}} & 8'h64)^
({8{data_in[0][7]}} & 8'hc8)^
({8{data_in[1][0]}} & 8'hdf)^
({8{data_in[1][1]}} & 8'h95)^
({8{data_in[1][2]}} & 8'h1)^
({8{data_in[1][3]}} & 8'h2)^
({8{data_in[1][4]}} & 8'h4)^
({8{data_in[1][5]}} & 8'h8)^
({8{data_in[1][6]}} & 8'h10)^
({8{data_in[1][7]}} & 8'h20)^
({8{data_in[2][0]}} & 8'h14)^
({8{data_in[2][1]}} & 8'h28)^
({8{data_in[2][2]}} & 8'h50)^
({8{data_in[2][3]}} & 8'ha0)^
({8{data_in[2][4]}} & 8'h6b)^
({8{data_in[2][5]}} & 8'hd6)^
({8{data_in[2][6]}} & 8'h87)^
({8{data_in[2][7]}} & 8'h25)^
({8{data_in[3][0]}} & 8'h79)^
({8{data_in[3][1]}} & 8'hf2)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[3][2]}} & 8'hcf)^
({{8{data_in[3][3]}} & 8'hb5)^
({{8{data_in[3][4]}} & 8'h41)^
({{8{data_in[3][5]}} & 8'h82)^
({{8{data_in[3][6]}} & 8'h2f)^
({{8{data_in[3][7]}} & 8'h5e)^
({{8{data_in[4][0]}} & 8'h92)^
({{8{data_in[4][1]}} & 8'hf)^
({{8{data_in[4][2]}} & 8'h1e)^
({{8{data_in[4][3]}} & 8'h3c)^
({{8{data_in[4][4]}} & 8'h78)^
({{8{data_in[4][5]}} & 8'hf0)^
({{8{data_in[4][6]}} & 8'hcb)^
({{8{data_in[4][7]}} & 8'hbd)^
({{8{data_in[5][0]}} & 8'h62)^
({{8{data_in[5][1]}} & 8'hc4)^
({{8{data_in[5][2]}} & 8'ha3)^
({{8{data_in[5][3]}} & 8'h6d)^
({{8{data_in[5][4]}} & 8'hda)^
({{8{data_in[5][5]}} & 8'h9f)^
({{8{data_in[5][6]}} & 8'h15)^
({{8{data_in[5][7]}} & 8'h2a)^
({{8{data_in[6][0]}} & 8'ha9)^
({{8{data_in[6][1]}} & 8'h79)^
({{8{data_in[6][2]}} & 8'hf2)^
({{8{data_in[6][3]}} & 8'hcf)^
({{8{data_in[6][4]}} & 8'hb5)^
({{8{data_in[6][5]}} & 8'h41)^
({{8{data_in[6][6]}} & 8'h82)^
({{8{data_in[6][7]}} & 8'h2f)^
({{8{data_in[7][0]}} & 8'hec)^
({{8{data_in[7][1]}} & 8'hf3)^
({{8{data_in[7][2]}} & 8'hcd)^
({{8{data_in[7][3]}} & 8'hb1)^
({{8{data_in[7][4]}} & 8'h49)^
({{8{data_in[7][5]}} & 8'h92)^
({{8{data_in[7][6]}} & 8'hf)^
({{8{data_in[7][7]}} & 8'h1e)^
({{8{data_in[8][0]}} & 8'hbb)^
({{8{data_in[8][1]}} & 8'h5d)^
({{8{data_in[8][2]}} & 8'hba)^
({{8{data_in[8][3]}} & 8'h5f)^
({{8{data_in[8][4]}} & 8'hbe)^
({{8{data_in[8][5]}} & 8'h57)^
({{8{data_in[8][6]}} & 8'hae)^
({{8{data_in[8][7]}} & 8'h77)^
({{8{data_in[9][0]}} & 8'h37)^
({{8{data_in[9][1]}} & 8'h6e)^
({{8{data_in[9][2]}} & 8'hdc)^
({{8{data_in[9][3]}} & 8'h93)^
({{8{data_in[9][4]}} & 8'hd)^
({{8{data_in[9][5]}} & 8'h1a)^
({{8{data_in[9][6]}} & 8'h34)^
({{8{data_in[9][7]}} & 8'h68)^
({{8{data_in[10][0]}} & 8'h52)^
({{8{data_in[10][1]}} & 8'ha4)^
({{8{data_in[10][2]}} & 8'h63)^
({{8{data_in[10][3]}} & 8'hc6)^
({{8{data_in[10][4]}} & 8'ha7)^
({{8{data_in[10][5]}} & 8'h65)^
({{8{data_in[10][6]}} & 8'hca)^
({{8{data_in[10][7]}} & 8'hbf)^
({{8{data_in[11][0]}} & 8'hca)^
({{8{data_in[11][1]}} & 8'hbf)^
({{8{data_in[11][2]}} & 8'h55)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[11][3]}} & 8'haa)^
({{8{data_in[11][4]}} & 8'h7f)^
({{8{data_in[11][5]}} & 8'hfe)^
({{8{data_in[11][6]}} & 8'hd7)^
({{8{data_in[11][7]}} & 8'h85)^
({{8{data_in[12][0]}} & 8'hd1)^
({{8{data_in[12][1]}} & 8'h89)^
({{8{data_in[12][2]}} & 8'h39)^
({{8{data_in[12][3]}} & 8'h72)^
({{8{data_in[12][4]}} & 8'he4)^
({{8{data_in[12][5]}} & 8'he3)^
({{8{data_in[12][6]}} & 8'hed)^
({{8{data_in[12][7]}} & 8'hf1)^
({{8{data_in[13][0]}} & 8'h79)^
({{8{data_in[13][1]}} & 8'hf2)^
({{8{data_in[13][2]}} & 8'hcf)^
({{8{data_in[13][3]}} & 8'hb5)^
({{8{data_in[13][4]}} & 8'h41)^
({{8{data_in[13][5]}} & 8'h82)^
({{8{data_in[13][6]}} & 8'h2f)^
({{8{data_in[13][7]}} & 8'h5e)^
({{8{data_in[14][0]}} & 8'h99)^
({{8{data_in[14][1]}} & 8'h19)^
({{8{data_in[14][2]}} & 8'h32)^
({{8{data_in[14][3]}} & 8'h64)^
({{8{data_in[14][4]}} & 8'hc8)^
({{8{data_in[14][5]}} & 8'hbb)^
({{8{data_in[14][6]}} & 8'h5d)^
({{8{data_in[14][7]}} & 8'hba)^
({{8{data_in[15][0]}} & 8'he2)^
({{8{data_in[15][1]}} & 8'hef)^
({{8{data_in[15][2]}} & 8'hf5)^
({{8{data_in[15][3]}} & 8'hc1)^
({{8{data_in[15][4]}} & 8'ha9)^
({{8{data_in[15][5]}} & 8'h79)^
({{8{data_in[15][6]}} & 8'hf2)^
({{8{data_in[15][7]}} & 8'hcf)^
({{8{data_in[16][0]}} & 8'hac)^
({{8{data_in[16][1]}} & 8'h73)^
({{8{data_in[16][2]}} & 8'he6)^
({{8{data_in[16][3]}} & 8'he7)^
({{8{data_in[16][4]}} & 8'he5)^
({{8{data_in[16][5]}} & 8'he1)^
({{8{data_in[16][6]}} & 8'he9)^
({{8{data_in[16][7]}} & 8'hf9)^
({{8{data_in[17][0]}} & 8'hcf)^
({{8{data_in[17][1]}} & 8'hb5)^
({{8{data_in[17][2]}} & 8'h41)^
({{8{data_in[17][3]}} & 8'h82)^
({{8{data_in[17][4]}} & 8'h2f)^
({{8{data_in[17][5]}} & 8'h5e)^
({{8{data_in[17][6]}} & 8'hbc)^
({{8{data_in[17][7]}} & 8'h53)^
({{8{data_in[18][0]}} & 8'h66)^
({{8{data_in[18][1]}} & 8'hcc)^
({{8{data_in[18][2]}} & 8'hb3)^
({{8{data_in[18][3]}} & 8'h4d)^
({{8{data_in[18][4]}} & 8'h9a)^
({{8{data_in[18][5]}} & 8'h1f)^
({{8{data_in[18][6]}} & 8'h3e)^
({{8{data_in[18][7]}} & 8'h7c)^
({{8{data_in[19][0]}} & 8'h9d)^
({{8{data_in[19][1]}} & 8'h11)^
({{8{data_in[19][2]}} & 8'h22)^
({{8{data_in[19][3]}} & 8'h44})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[19][4]}} & 8'h88)^
({{8{data_in[19][5]}} & 8'h3b)^
({{8{data_in[19][6]}} & 8'h76)^
({{8{data_in[19][7]}} & 8'hec)^
({{8{data_in[20][0]}} & 8'h4f)^
({{8{data_in[20][1]}} & 8'h9e)^
({{8{data_in[20][2]}} & 8'h17)^
({{8{data_in[20][3]}} & 8'h2e)^
({{8{data_in[20][4]}} & 8'h5c)^
({{8{data_in[20][5]}} & 8'hb8)^
({{8{data_in[20][6]}} & 8'h5b)^
({{8{data_in[20][7]}} & 8'hb6)^
({{8{data_in[21][0]}} & 8'ha4)^
({{8{data_in[21][1]}} & 8'h63)^
({{8{data_in[21][2]}} & 8'hc6)^
({{8{data_in[21][3]}} & 8'ha7)^
({{8{data_in[21][4]}} & 8'h65)^
({{8{data_in[21][5]}} & 8'hca)^
({{8{data_in[21][6]}} & 8'hbf)^
({{8{data_in[21][7]}} & 8'h55)^
({{8{data_in[22][0]}} & 8'hb2)^
({{8{data_in[22][1]}} & 8'h4f)^
({{8{data_in[22][2]}} & 8'h9e)^
({{8{data_in[22][3]}} & 8'h17)^
({{8{data_in[22][4]}} & 8'h2e)^
({{8{data_in[22][5]}} & 8'h5c)^
({{8{data_in[22][6]}} & 8'hb8)^
({{8{data_in[22][7]}} & 8'h5b)^
({{8{data_in[23][0]}} & 8'h94)^
({{8{data_in[23][1]}} & 8'h3)^
({{8{data_in[23][2]}} & 8'h6)^
({{8{data_in[23][3]}} & 8'hc)^
({{8{data_in[23][4]}} & 8'h18)^
({{8{data_in[23][5]}} & 8'h30)^
({{8{data_in[23][6]}} & 8'h60)^
({{8{data_in[23][7]}} & 8'hc0)^
({{8{data_in[24][0]}} & 8'ha9)^
({{8{data_in[24][1]}} & 8'h79)^
({{8{data_in[24][2]}} & 8'hf2)^
({{8{data_in[24][3]}} & 8'hcf)^
({{8{data_in[24][4]}} & 8'hb5)^
({{8{data_in[24][5]}} & 8'h41)^
({{8{data_in[24][6]}} & 8'h82)^
({{8{data_in[24][7]}} & 8'h2f)^
({{8{data_in[25][0]}} & 8'h14)^
({{8{data_in[25][1]}} & 8'h28)^
({{8{data_in[25][2]}} & 8'h50)^
({{8{data_in[25][3]}} & 8'ha0)^
({{8{data_in[25][4]}} & 8'h6b)^
({{8{data_in[25][5]}} & 8'hd6)^
({{8{data_in[25][6]}} & 8'h87)^
({{8{data_in[25][7]}} & 8'h25)^
({{8{data_in[26][0]}} & 8'h60)^
({{8{data_in[26][1]}} & 8'hc0)^
({{8{data_in[26][2]}} & 8'hab)^
({{8{data_in[26][3]}} & 8'h7d)^
({{8{data_in[26][4]}} & 8'hfa)^
({{8{data_in[26][5]}} & 8'hdf)^
({{8{data_in[26][6]}} & 8'h95)^
({{8{data_in[26][7]}} & 8'h1)^
({{8{data_in[27][0]}} & 8'he8)^
({{8{data_in[27][1]}} & 8'fib)^
({{8{data_in[27][2]}} & 8'hdd)^
({{8{data_in[27][3]}} & 8'h91)^
({{8{data_in[27][4]}} & 8'h9)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[27][5]}} & 8'h12)^
({{8{data_in[27][6]}} & 8'h24)^
({{8{data_in[27][7]}} & 8'h48)^
({{8{data_in[28][0]}} & 8'h5b)^
({{8{data_in[28][1]}} & 8'hb6)^
({{8{data_in[28][2]}} & 8'h47)^
({{8{data_in[28][3]}} & 8'h8e)^
({{8{data_in[28][4]}} & 8'h37)^
({{8{data_in[28][5]}} & 8'h6e)^
({{8{data_in[28][6]}} & 8'hdc)^
({{8{data_in[28][7]}} & 8'h93)^
({{8{data_in[29][0]}} & 8'hb)^
({{8{data_in[29][1]}} & 8'h16)^
({{8{data_in[29][2]}} & 8'h2c)^
({{8{data_in[29][3]}} & 8'h58)^
({{8{data_in[29][4]}} & 8'hb0)^
({{8{data_in[29][5]}} & 8'h4b)^
({{8{data_in[29][6]}} & 8'h96)^
({{8{data_in[29][7]}} & 8'h7)^
({{8{data_in[30][0]}} & 8'ha1)^
({{8{data_in[30][1]}} & 8'h69)^
({{8{data_in[30][2]}} & 8'hd2)^
({{8{data_in[30][3]}} & 8'h8f)^
({{8{data_in[30][4]}} & 8'h35)^
({{8{data_in[30][5]}} & 8'h6a)^
({{8{data_in[30][6]}} & 8'hd4)^
({{8{data_in[30][7]}} & 8'h83)^
({{8{data_in[31][0]}} & 8'h1c)^
({{8{data_in[31][1]}} & 8'h38)^
({{8{data_in[31][2]}} & 8'h70)^
({{8{data_in[31][3]}} & 8'he0)^
({{8{data_in[31][4]}} & 8'heb)^
({{8{data_in[31][5]}} & 8'hfd)^
({{8{data_in[31][6]}} & 8'hd1)^
({{8{data_in[31][7]}} & 8'h89)^
({{8{data_in[32][0]}} & 8'hf1)^
({{8{data_in[32][1]}} & 8'hc9)^
({{8{data_in[32][2]}} & 8'hb9)^
({{8{data_in[32][3]}} & 8'h59)^
({{8{data_in[32][4]}} & 8'hb2)^
({{8{data_in[32][5]}} & 8'h4f)^
({{8{data_in[32][6]}} & 8'h9e)^
({{8{data_in[32][7]}} & 8'h17)^
({{8{data_in[33][0]}} & 8'hde)^
({{8{data_in[33][1]}} & 8'h97)^
({{8{data_in[33][2]}} & 8'h5)^
({{8{data_in[33][3]}} & 8'ha)^
({{8{data_in[33][4]}} & 8'h14)^
({{8{data_in[33][5]}} & 8'h28)^
({{8{data_in[33][6]}} & 8'h50)^
({{8{data_in[33][7]}} & 8'ha0)^
({{8{data_in[34][0]}} & 8'h9b)^
({{8{data_in[34][1]}} & 8'h1d)^
({{8{data_in[34][2]}} & 8'h3a)^
({{8{data_in[34][3]}} & 8'h74)^
({{8{data_in[34][4]}} & 8'he8)^
({{8{data_in[34][5]}} & 8'hfb)^
({{8{data_in[34][6]}} & 8'hdd)^
({{8{data_in[34][7]}} & 8'h91)^
({{8{data_in[35][0]}} & 8'hed)^
({{8{data_in[35][1]}} & 8'hf1)^
({{8{data_in[35][2]}} & 8'hc9)^
({{8{data_in[35][3]}} & 8'hb9)^
({{8{data_in[35][4]}} & 8'h59)^
({{8{data_in[35][5]}} & 8'hb2)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[35][6]}} & 8'h4f)^
({{8{data_in[35][7]}} & 8'h9e)^
({{8{data_in[36][0]}} & 8'h88)^
({{8{data_in[36][1]}} & 8'h3b)^
({{8{data_in[36][2]}} & 8'h76)^
({{8{data_in[36][3]}} & 8'hec)^
({{8{data_in[36][4]}} & 8'hf3)^
({{8{data_in[36][5]}} & 8'hcd)^
({{8{data_in[36][6]}} & 8'hb1)^
({{8{data_in[36][7]}} & 8'h49)^
({{8{data_in[37][0]}} & 8'hcb)^
({{8{data_in[37][1]}} & 8'hbd)^
({{8{data_in[37][2]}} & 8'h51)^
({{8{data_in[37][3]}} & 8'ha2)^
({{8{data_in[37][4]}} & 8'h6f)^
({{8{data_in[37][5]}} & 8'hde)^
({{8{data_in[37][6]}} & 8'h97)^
({{8{data_in[37][7]}} & 8'h5)^
({{8{data_in[38][0]}} & 8'hce)^
({{8{data_in[38][1]}} & 8'hb7)^
({{8{data_in[38][2]}} & 8'h45)^
({{8{data_in[38][3]}} & 8'h8a)^
({{8{data_in[38][4]}} & 8'h3f)^
({{8{data_in[38][5]}} & 8'h7e)^
({{8{data_in[38][6]}} & 8'hfc)^
({{8{data_in[38][7]}} & 8'hd3)^
({{8{data_in[39][0]}} & 8'hae)^
({{8{data_in[39][1]}} & 8'h77)^
({{8{data_in[39][2]}} & 8'hee)^
({{8{data_in[39][3]}} & 8'hf7)^
({{8{data_in[39][4]}} & 8'hc5)^
({{8{data_in[39][5]}} & 8'ha1)^
({{8{data_in[39][6]}} & 8'h69)^
({{8{data_in[39][7]}} & 8'hd2)^
({{8{data_in[40][0]}} & 8'ha0)^
({{8{data_in[40][1]}} & 8'h6b)^
({{8{data_in[40][2]}} & 8'hd6)^
({{8{data_in[40][3]}} & 8'h87)^
({{8{data_in[40][4]}} & 8'h25)^
({{8{data_in[40][5]}} & 8'h4a)^
({{8{data_in[40][6]}} & 8'h94)^
({{8{data_in[40][7]}} & 8'h3)^
({{8{data_in[41][0]}} & 8'h33)^
({{8{data_in[41][1]}} & 8'h66)^
({{8{data_in[41][2]}} & 8'hcc)^
({{8{data_in[41][3]}} & 8'hb3)^
({{8{data_in[41][4]}} & 8'h4d)^
({{8{data_in[41][5]}} & 8'h9a)^
({{8{data_in[41][6]}} & 8'h1f)^
({{8{data_in[41][7]}} & 8'h3e)^
({{8{data_in[42][0]}} & 8'h38)^
({{8{data_in[42][1]}} & 8'h70)^
({{8{data_in[42][2]}} & 8'he0)^
({{8{data_in[42][3]}} & 8'heb)^
({{8{data_in[42][4]}} & 8'hfd)^
({{8{data_in[42][5]}} & 8'hd1)^
({{8{data_in[42][6]}} & 8'h89)^
({{8{data_in[42][7]}} & 8'h39)^
({{8{data_in[43][0]}} & 8'h50)^
({{8{data_in[43][1]}} & 8'ha0)^
({{8{data_in[43][2]}} & 8'h6b)^
({{8{data_in[43][3]}} & 8'hd6)^
({{8{data_in[43][4]}} & 8'h87)^
({{8{data_in[43][5]}} & 8'h25)^
({{8{data_in[43][6]}} & 8'h4a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[43][7]}} & 8'h94)^
({{8{data_in[44][0]}} & 8'h22)^
({{8{data_in[44][1]}} & 8'h44)^
({{8{data_in[44][2]}} & 8'h88)^
({{8{data_in[44][3]}} & 8'h3b)^
({{8{data_in[44][4]}} & 8'h76)^
({{8{data_in[44][5]}} & 8'hec)^
({{8{data_in[44][6]}} & 8'hf3)^
({{8{data_in[44][7]}} & 8'hcd)^
({{8{data_in[45][0]}} & 8'hd5)^
({{8{data_in[45][1]}} & 8'h81)^
({{8{data_in[45][2]}} & 8'h29)^
({{8{data_in[45][3]}} & 8'h52)^
({{8{data_in[45][4]}} & 8'ha4)^
({{8{data_in[45][5]}} & 8'h63)^
({{8{data_in[45][6]}} & 8'hc6)^
({{8{data_in[45][7]}} & 8'ha7)^
({{8{data_in[46][0]}} & 8'h43)^
({{8{data_in[46][1]}} & 8'h86)^
({{8{data_in[46][2]}} & 8'h27)^
({{8{data_in[46][3]}} & 8'h4e)^
({{8{data_in[46][4]}} & 8'h9c)^
({{8{data_in[46][5]}} & 8'h13)^
({{8{data_in[46][6]}} & 8'h26)^
({{8{data_in[46][7]}} & 8'h4c)^
({{8{data_in[47][0]}} & 8'h5d)^
({{8{data_in[47][1]}} & 8'hba)^
({{8{data_in[47][2]}} & 8'h5f)^
({{8{data_in[47][3]}} & 8'hbe)^
({{8{data_in[47][4]}} & 8'h57)^
({{8{data_in[47][5]}} & 8'hae)^
({{8{data_in[47][6]}} & 8'h77)^
({{8{data_in[47][7]}} & 8'hee)^
({{8{data_in[48][0]}} & 8'h61)^
({{8{data_in[48][1]}} & 8'hc2)^
({{8{data_in[48][2]}} & 8'haf)^
({{8{data_in[48][3]}} & 8'h75)^
({{8{data_in[48][4]}} & 8'hea)^
({{8{data_in[48][5]}} & 8'hff)^
({{8{data_in[48][6]}} & 8'hd5)^
({{8{data_in[48][7]}} & 8'h81)^
({{8{data_in[49][0]}} & 8'hef)^
({{8{data_in[49][1]}} & 8'hf5)^
({{8{data_in[49][2]}} & 8'hc1)^
({{8{data_in[49][3]}} & 8'ha9)^
({{8{data_in[49][4]}} & 8'h79)^
({{8{data_in[49][5]}} & 8'hf2)^
({{8{data_in[49][6]}} & 8'hcf)^
({{8{data_in[49][7]}} & 8'hb5)^
({{8{data_in[50][0]}} & 8'h53)^
({{8{data_in[50][1]}} & 8'ha6)^
({{8{data_in[50][2]}} & 8'h67)^
({{8{data_in[50][3]}} & 8'hce)^
({{8{data_in[50][4]}} & 8'hb7)^
({{8{data_in[50][5]}} & 8'h45)^
({{8{data_in[50][6]}} & 8'h8a)^
({{8{data_in[50][7]}} & 8'h3f)^
({{8{data_in[51][0]}} & 8'h45)^
({{8{data_in[51][1]}} & 8'h8a)^
({{8{data_in[51][2]}} & 8'h3f)^
({{8{data_in[51][3]}} & 8'h7e)^
({{8{data_in[51][4]}} & 8'hfc)^
({{8{data_in[51][5]}} & 8'hd3)^
({{8{data_in[51][6]}} & 8'h8d)^
({{8{data_in[51][7]}} & 8'h31)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[52][0]}} & 8'h41)^
({{8{data_in[52][1]}} & 8'h82)^
({{8{data_in[52][2]}} & 8'h2f)^
({{8{data_in[52][3]}} & 8'h5e)^
({{8{data_in[52][4]}} & 8'hbc)^
({{8{data_in[52][5]}} & 8'h53)^
({{8{data_in[52][6]}} & 8'ha6)^
({{8{data_in[52][7]}} & 8'h67)^
({{8{data_in[53][0]}} & 8'h4b)^
({{8{data_in[53][1]}} & 8'h96)^
({{8{data_in[53][2]}} & 8'h7)^
({{8{data_in[53][3]}} & 8'he)^
({{8{data_in[53][4]}} & 8'h1c)^
({{8{data_in[53][5]}} & 8'h38)^
({{8{data_in[53][6]}} & 8'h70)^
({{8{data_in[53][7]}} & 8'he0)^
({{8{data_in[54][0]}} & 8'h3f)^
({{8{data_in[54][1]}} & 8'h7e)^
({{8{data_in[54][2]}} & 8'hfc)^
({{8{data_in[54][3]}} & 8'hd3)^
({{8{data_in[54][4]}} & 8'h8d)^
({{8{data_in[54][5]}} & 8'h31)^
({{8{data_in[54][6]}} & 8'h62)^
({{8{data_in[54][7]}} & 8'hc4)^
({{8{data_in[55][0]}} & 8'hc0)^
({{8{data_in[55][1]}} & 8'hab)^
({{8{data_in[55][2]}} & 8'h7d)^
({{8{data_in[55][3]}} & 8'hfa)^
({{8{data_in[55][4]}} & 8'hdf)^
({{8{data_in[55][5]}} & 8'h95)^
({{8{data_in[55][6]}} & 8'h1)^
({{8{data_in[55][7]}} & 8'h2)^
({{8{data_in[56][0]}} & 8'h1b)^
({{8{data_in[56][1]}} & 8'h36)^
({{8{data_in[56][2]}} & 8'h6c)^
({{8{data_in[56][3]}} & 8'hd8)^
({{8{data_in[56][4]}} & 8'h9b)^
({{8{data_in[56][5]}} & 8'h1d)^
({{8{data_in[56][6]}} & 8'h3a)^
({{8{data_in[56][7]}} & 8'h74)^
({{8{data_in[57][0]}} & 8'hf6)^
({{8{data_in[57][1]}} & 8'hc7)^
({{8{data_in[57][2]}} & 8'ha5)^
({{8{data_in[57][3]}} & 8'h61)^
({{8{data_in[57][4]}} & 8'hc2)^
({{8{data_in[57][5]}} & 8'haf)^
({{8{data_in[57][6]}} & 8'h75)^
({{8{data_in[57][7]}} & 8'hea)^
({{8{data_in[58][0]}} & 8'h9f)^
({{8{data_in[58][1]}} & 8'h15)^
({{8{data_in[58][2]}} & 8'h2a)^
({{8{data_in[58][3]}} & 8'h54)^
({{8{data_in[58][4]}} & 8'ha8)^
({{8{data_in[58][5]}} & 8'h7b)^
({{8{data_in[58][6]}} & 8'hf6)^
({{8{data_in[58][7]}} & 8'hc7)^
({{8{data_in[59][0]}} & 8'h33)^
({{8{data_in[59][1]}} & 8'h66)^
({{8{data_in[59][2]}} & 8'hcc)^
({{8{data_in[59][3]}} & 8'hb3)^
({{8{data_in[59][4]}} & 8'h4d)^
({{8{data_in[59][5]}} & 8'h9a)^
({{8{data_in[59][6]}} & 8'h1f)^
({{8{data_in[59][7]}} & 8'h3e)^
({{8{data_in[60][0]}} & 8'h9a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[60][1]}} & 8'h1f)^
({{8{data_in[60][2]}} & 8'h3e)^
({{8{data_in[60][3]}} & 8'h7c)^
({{8{data_in[60][4]}} & 8'hf8)^
({{8{data_in[60][5]}} & 8'hdb)^
({{8{data_in[60][6]}} & 8'h9d)^
({{8{data_in[60][7]}} & 8'h11)^
({{8{data_in[61][0]}} & 8'h1d)^
({{8{data_in[61][1]}} & 8'h3a)^
({{8{data_in[61][2]}} & 8'h74)^
({{8{data_in[61][3]}} & 8'he8)^
({{8{data_in[61][4]}} & 8'hfb)^
({{8{data_in[61][5]}} & 8'hdd)^
({{8{data_in[61][6]}} & 8'h91)^
({{8{data_in[61][7]}} & 8'h9)^
({{8{data_in[62][0]}} & 8'h50)^
({{8{data_in[62][1]}} & 8'ha0)^
({{8{data_in[62][2]}} & 8'h6b)^
({{8{data_in[62][3]}} & 8'hd6)^
({{8{data_in[62][4]}} & 8'h87)^
({{8{data_in[62][5]}} & 8'h25)^
({{8{data_in[62][6]}} & 8'h4a)^
({{8{data_in[62][7]}} & 8'h94)^
({{8{data_in[63][0]}} & 8'he7)^
({{8{data_in[63][1]}} & 8'he5)^
({{8{data_in[63][2]}} & 8'he1)^
({{8{data_in[63][3]}} & 8'he9)^
({{8{data_in[63][4]}} & 8'hf9)^
({{8{data_in[63][5]}} & 8'hd9)^
({{8{data_in[63][6]}} & 8'h99)^
({{8{data_in[63][7]}} & 8'h19)^
({{8{data_in[64][0]}} & 8'hfb)^
({{8{data_in[64][1]}} & 8'hdd)^
({{8{data_in[64][2]}} & 8'h91)^
({{8{data_in[64][3]}} & 8'h9)^
({{8{data_in[64][4]}} & 8'h12)^
({{8{data_in[64][5]}} & 8'h24)^
({{8{data_in[64][6]}} & 8'h48)^
({{8{data_in[64][7]}} & 8'h90)^
({{8{data_in[65][0]}} & 8'he1)^
({{8{data_in[65][1]}} & 8'he9)^
({{8{data_in[65][2]}} & 8'hf9)^
({{8{data_in[65][3]}} & 8'hd9)^
({{8{data_in[65][4]}} & 8'h99)^
({{8{data_in[65][5]}} & 8'h19)^
({{8{data_in[65][6]}} & 8'h32)^
({{8{data_in[65][7]}} & 8'h64)^
({{8{data_in[66][0]}} & 8'h91)^
({{8{data_in[66][1]}} & 8'h9)^
({{8{data_in[66][2]}} & 8'h12)^
({{8{data_in[66][3]}} & 8'h24)^
({{8{data_in[66][4]}} & 8'h48)^
({{8{data_in[66][5]}} & 8'h90)^
({{8{data_in[66][6]}} & 8'hb)^
({{8{data_in[66][7]}} & 8'h16)^
({{8{data_in[67][0]}} & 8'h19)^
({{8{data_in[67][1]}} & 8'h32)^
({{8{data_in[67][2]}} & 8'h64)^
({{8{data_in[67][3]}} & 8'hc8)^
({{8{data_in[67][4]}} & 8'hbb)^
({{8{data_in[67][5]}} & 8'h5d)^
({{8{data_in[67][6]}} & 8'hba)^
({{8{data_in[67][7]}} & 8'h5f)^
({{8{data_in[68][0]}} & 8'h6b)^
({{8{data_in[68][1]}} & 8'hd6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[68][2]}} & 8'h87)^
({{8{data_in[68][3]}} & 8'h25)^
({{8{data_in[68][4]}} & 8'h4a)^
({{8{data_in[68][5]}} & 8'h94)^
({{8{data_in[68][6]}} & 8'h3)^
({{8{data_in[68][7]}} & 8'h6)^
({{8{data_in[69][0]}} & 8'h40)^
({{8{data_in[69][1]}} & 8'h80)^
({{8{data_in[69][2]}} & 8'h2b)^
({{8{data_in[69][3]}} & 8'h56)^
({{8{data_in[69][4]}} & 8'hac)^
({{8{data_in[69][5]}} & 8'h73)^
({{8{data_in[69][6]}} & 8'he6)^
({{8{data_in[69][7]}} & 8'he7)^
({{8{data_in[70][0]}} & 8'h9c)^
({{8{data_in[70][1]}} & 8'h13)^
({{8{data_in[70][2]}} & 8'h26)^
({{8{data_in[70][3]}} & 8'h4c)^
({{8{data_in[70][4]}} & 8'h98)^
({{8{data_in[70][5]}} & 8'h1b)^
({{8{data_in[70][6]}} & 8'h36)^
({{8{data_in[70][7]}} & 8'h6c)^
({{8{data_in[71][0]}} & 8'h1)^
({{8{data_in[71][1]}} & 8'h2)^
({{8{data_in[71][2]}} & 8'h4)^
({{8{data_in[71][3]}} & 8'h8)^
({{8{data_in[71][4]}} & 8'h10)^
({{8{data_in[71][5]}} & 8'h20)^
({{8{data_in[71][6]}} & 8'h40)^
({{8{data_in[71][7]}} & 8'h80)^
({{8{data_in[72][0]}} & 8'h7e)^
({{8{data_in[72][1]}} & 8'hfc)^
({{8{data_in[72][2]}} & 8'hd3)^
({{8{data_in[72][3]}} & 8'h8d)^
({{8{data_in[72][4]}} & 8'h31)^
({{8{data_in[72][5]}} & 8'h62)^
({{8{data_in[72][6]}} & 8'hc4)^
({{8{data_in[72][7]}} & 8'ha3)^
({{8{data_in[73][0]}} & 8'h64)^
({{8{data_in[73][1]}} & 8'hc8)^
({{8{data_in[73][2]}} & 8'hbb)^
({{8{data_in[73][3]}} & 8'h5d)^
({{8{data_in[73][4]}} & 8'hba)^
({{8{data_in[73][5]}} & 8'h5f)^
({{8{data_in[73][6]}} & 8'hbe)^
({{8{data_in[73][7]}} & 8'h57)^
({{8{data_in[74][0]}} & 8'h7d)^
({{8{data_in[74][1]}} & 8'hfa)^
({{8{data_in[74][2]}} & 8'hdf)^
({{8{data_in[74][3]}} & 8'h95)^
({{8{data_in[74][4]}} & 8'h1)^
({{8{data_in[74][5]}} & 8'h2)^
({{8{data_in[74][6]}} & 8'h4)^
({{8{data_in[74][7]}} & 8'h8)^
({{8{data_in[75][0]}} & 8'h7)^
({{8{data_in[75][1]}} & 8'he)^
({{8{data_in[75][2]}} & 8'h1c)^
({{8{data_in[75][3]}} & 8'h38)^
({{8{data_in[75][4]}} & 8'h70)^
({{8{data_in[75][5]}} & 8'he0)^
({{8{data_in[75][6]}} & 8'heb)^
({{8{data_in[75][7]}} & 8'hfd)^
({{8{data_in[76][0]}} & 8'h34)^
({{8{data_in[76][1]}} & 8'h68)^
({{8{data_in[76][2]}} & 8'hd0})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[76][3]}} & 8'h8b)^
({{8{data_in[76][4]}} & 8'h3d)^
({{8{data_in[76][5]}} & 8'h7a)^
({{8{data_in[76][6]}} & 8'hf4)^
({{8{data_in[76][7]}} & 8'hc3)^
({{8{data_in[77][0]}} & 8'h1d)^
({{8{data_in[77][1]}} & 8'h3a)^
({{8{data_in[77][2]}} & 8'h74)^
({{8{data_in[77][3]}} & 8'he8)^
({{8{data_in[77][4]}} & 8'hfb)^
({{8{data_in[77][5]}} & 8'hdd)^
({{8{data_in[77][6]}} & 8'h91)^
({{8{data_in[77][7]}} & 8'h9)^
({{8{data_in[78][0]}} & 8'hb8)^
({{8{data_in[78][1]}} & 8'h5b)^
({{8{data_in[78][2]}} & 8'hb6)^
({{8{data_in[78][3]}} & 8'h47)^
({{8{data_in[78][4]}} & 8'h8e)^
({{8{data_in[78][5]}} & 8'h37)^
({{8{data_in[78][6]}} & 8'h6e)^
({{8{data_in[78][7]}} & 8'hdc)^
({{8{data_in[79][0]}} & 8'h7)^
({{8{data_in[79][1]}} & 8'he)^
({{8{data_in[79][2]}} & 8'h1c)^
({{8{data_in[79][3]}} & 8'h38)^
({{8{data_in[79][4]}} & 8'h70)^
({{8{data_in[79][5]}} & 8'he0)^
({{8{data_in[79][6]}} & 8'heb)^
({{8{data_in[79][7]}} & 8'hfd)^
({{8{data_in[80][0]}} & 8'h7f)^
({{8{data_in[80][1]}} & 8'hfe)^
({{8{data_in[80][2]}} & 8'hd7)^
({{8{data_in[80][3]}} & 8'h85)^
({{8{data_in[80][4]}} & 8'h21)^
({{8{data_in[80][5]}} & 8'h42)^
({{8{data_in[80][6]}} & 8'h84)^
({{8{data_in[80][7]}} & 8'h23)^
({{8{data_in[81][0]}} & 8'hc0)^
({{8{data_in[81][1]}} & 8'hab)^
({{8{data_in[81][2]}} & 8'h7d)^
({{8{data_in[81][3]}} & 8'hfa)^
({{8{data_in[81][4]}} & 8'hdf)^
({{8{data_in[81][5]}} & 8'h95)^
({{8{data_in[81][6]}} & 8'h1)^
({{8{data_in[81][7]}} & 8'h2)^
({{8{data_in[82][0]}} & 8'h69)^
({{8{data_in[82][1]}} & 8'hd2)^
({{8{data_in[82][2]}} & 8'h8f)^
({{8{data_in[82][3]}} & 8'h35)^
({{8{data_in[82][4]}} & 8'h6a)^
({{8{data_in[82][5]}} & 8'hd4)^
({{8{data_in[82][6]}} & 8'h83)^
({{8{data_in[82][7]}} & 8'h2d)^
({{8{data_in[83][0]}} & 8'h39)^
({{8{data_in[83][1]}} & 8'h72)^
({{8{data_in[83][2]}} & 8'he4)^
({{8{data_in[83][3]}} & 8'he3)^
({{8{data_in[83][4]}} & 8'hed)^
({{8{data_in[83][5]}} & 8'hf1)^
({{8{data_in[83][6]}} & 8'hc9)^
({{8{data_in[83][7]}} & 8'hb9)^
({{8{data_in[84][0]}} & 8'hfb)^
({{8{data_in[84][1]}} & 8'hdd)^
({{8{data_in[84][2]}} & 8'h91)^
({{8{data_in[84][3]}} & 8'h9)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[84][4]}} & 8'h12)^
({{8{data_in[84][5]}} & 8'h24)^
({{8{data_in[84][6]}} & 8'h48)^
({{8{data_in[84][7]}} & 8'h90)^
({{8{data_in[85][0]}} & 8'h85)^
({{8{data_in[85][1]}} & 8'h21)^
({{8{data_in[85][2]}} & 8'h42)^
({{8{data_in[85][3]}} & 8'h84)^
({{8{data_in[85][4]}} & 8'h23)^
({{8{data_in[85][5]}} & 8'h46)^
({{8{data_in[85][6]}} & 8'h8c)^
({{8{data_in[85][7]}} & 8'h33)^
({{8{data_in[86][0]}} & 8'h41)^
({{8{data_in[86][1]}} & 8'h82)^
({{8{data_in[86][2]}} & 8'h2f)^
({{8{data_in[86][3]}} & 8'h5e)^
({{8{data_in[86][4]}} & 8'hbc)^
({{8{data_in[86][5]}} & 8'h53)^
({{8{data_in[86][6]}} & 8'ha6)^
({{8{data_in[86][7]}} & 8'h67)^
({{8{data_in[87][0]}} & 8'hd7)^
({{8{data_in[87][1]}} & 8'h85)^
({{8{data_in[87][2]}} & 8'h21)^
({{8{data_in[87][3]}} & 8'h42)^
({{8{data_in[87][4]}} & 8'h84)^
({{8{data_in[87][5]}} & 8'h23)^
({{8{data_in[87][6]}} & 8'h46)^
({{8{data_in[87][7]}} & 8'h8c)^
({{8{data_in[88][0]}} & 8'hff)^
({{8{data_in[88][1]}} & 8'hd5)^
({{8{data_in[88][2]}} & 8'h81)^
({{8{data_in[88][3]}} & 8'h29)^
({{8{data_in[88][4]}} & 8'h52)^
({{8{data_in[88][5]}} & 8'ha4)^
({{8{data_in[88][6]}} & 8'h63)^
({{8{data_in[88][7]}} & 8'hc6)^
({{8{data_in[89][0]}} & 8'hbd)^
({{8{data_in[89][1]}} & 8'h51)^
({{8{data_in[89][2]}} & 8'ha2)^
({{8{data_in[89][3]}} & 8'h6f)^
({{8{data_in[89][4]}} & 8'hde)^
({{8{data_in[89][5]}} & 8'h97)^
({{8{data_in[89][6]}} & 8'h5)^
({{8{data_in[89][7]}} & 8'ha)^
({{8{data_in[90][0]}} & 8'hdc)^
({{8{data_in[90][1]}} & 8'h93)^
({{8{data_in[90][2]}} & 8'hd)^
({{8{data_in[90][3]}} & 8'h1a)^
({{8{data_in[90][4]}} & 8'h34)^
({{8{data_in[90][5]}} & 8'h68)^
({{8{data_in[90][6]}} & 8'hd0)^
({{8{data_in[90][7]}} & 8'h8b)^
({{8{data_in[91][0]}} & 8'ha7)^
({{8{data_in[91][1]}} & 8'h65)^
({{8{data_in[91][2]}} & 8'hca)^
({{8{data_in[91][3]}} & 8'hbf)^
({{8{data_in[91][4]}} & 8'h55)^
({{8{data_in[91][5]}} & 8'haa)^
({{8{data_in[91][6]}} & 8'h7f)^
({{8{data_in[91][7]}} & 8'hfe)^
({{8{data_in[92][0]}} & 8'h4b)^
({{8{data_in[92][1]}} & 8'h96)^
({{8{data_in[92][2]}} & 8'h7)^
({{8{data_in[92][3]}} & 8'he)^
({{8{data_in[92][4]}} & 8'h1c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[92][5]}} & 8'h38)^
({{8{data_in[92][6]}} & 8'h70)^
({{8{data_in[92][7]}} & 8'he0)^
({{8{data_in[93][0]}} & 8'he2)^
({{8{data_in[93][1]}} & 8'hef)^
({{8{data_in[93][2]}} & 8'hf5)^
({{8{data_in[93][3]}} & 8'hc1)^
({{8{data_in[93][4]}} & 8'ha9)^
({{8{data_in[93][5]}} & 8'h79)^
({{8{data_in[93][6]}} & 8'hf2)^
({{8{data_in[93][7]}} & 8'hcf)^
({{8{data_in[94][0]}} & 8'h42)^
({{8{data_in[94][1]}} & 8'h84)^
({{8{data_in[94][2]}} & 8'h23)^
({{8{data_in[94][3]}} & 8'h46)^
({{8{data_in[94][4]}} & 8'h8c)^
({{8{data_in[94][5]}} & 8'h33)^
({{8{data_in[94][6]}} & 8'h66)^
({{8{data_in[94][7]}} & 8'hcc)^
({{8{data_in[95][0]}} & 8'h34)^
({{8{data_in[95][1]}} & 8'h68)^
({{8{data_in[95][2]}} & 8'hd0)^
({{8{data_in[95][3]}} & 8'h8b)^
({{8{data_in[95][4]}} & 8'h3d)^
({{8{data_in[95][5]}} & 8'h7a)^
({{8{data_in[95][6]}} & 8'hf4)^
({{8{data_in[95][7]}} & 8'hc3)^
({{8{data_in[96][0]}} & 8'h18)^
({{8{data_in[96][1]}} & 8'h30)^
({{8{data_in[96][2]}} & 8'h60)^
({{8{data_in[96][3]}} & 8'hc0)^
({{8{data_in[96][4]}} & 8'hab)^
({{8{data_in[96][5]}} & 8'h7d)^
({{8{data_in[96][6]}} & 8'hfa)^
({{8{data_in[96][7]}} & 8'hdf)^
({{8{data_in[97][0]}} & 8'h5d)^
({{8{data_in[97][1]}} & 8'hba)^
({{8{data_in[97][2]}} & 8'h5f)^
({{8{data_in[97][3]}} & 8'hbe)^
({{8{data_in[97][4]}} & 8'h57)^
({{8{data_in[97][5]}} & 8'hae)^
({{8{data_in[97][6]}} & 8'h77)^
({{8{data_in[97][7]}} & 8'hee)^
({{8{data_in[98][0]}} & 8'h76)^
({{8{data_in[98][1]}} & 8'hec)^
({{8{data_in[98][2]}} & 8'hf3)^
({{8{data_in[98][3]}} & 8'hcd)^
({{8{data_in[98][4]}} & 8'hb1)^
({{8{data_in[98][5]}} & 8'h49)^
({{8{data_in[98][6]}} & 8'h92)^
({{8{data_in[98][7]}} & 8'hf)^
({{8{data_in[99][0]}} & 8'h4c)^
({{8{data_in[99][1]}} & 8'h98)^
({{8{data_in[99][2]}} & 8'h1b)^
({{8{data_in[99][3]}} & 8'h36)^
({{8{data_in[99][4]}} & 8'h6c)^
({{8{data_in[99][5]}} & 8'hd8)^
({{8{data_in[99][6]}} & 8'h9b)^
({{8{data_in[99][7]}} & 8'h1d)^
({{8{data_in[100][0]}} & 8'h2c)^
({{8{data_in[100][1]}} & 8'h58)^
({{8{data_in[100][2]}} & 8'hb0)^
({{8{data_in[100][3]}} & 8'h4b)^
({{8{data_in[100][4]}} & 8'h96)^
({{8{data_in[100][5]}} & 8'h7)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[100][6]}} & 8'he)^
({{8{data_in[100][7]}} & 8'h1c)^
({{8{data_in[101][0]}} & 8'h67)^
({{8{data_in[101][1]}} & 8'hce)^
({{8{data_in[101][2]}} & 8'hb7)^
({{8{data_in[101][3]}} & 8'h45)^
({{8{data_in[101][4]}} & 8'h8a)^
({{8{data_in[101][5]}} & 8'h3f)^
({{8{data_in[101][6]}} & 8'h7e)^
({{8{data_in[101][7]}} & 8'hfc)^
({{8{data_in[102][0]}} & 8'h1c)^
({{8{data_in[102][1]}} & 8'h38)^
({{8{data_in[102][2]}} & 8'h70)^
({{8{data_in[102][3]}} & 8'he0)^
({{8{data_in[102][4]}} & 8'heb)^
({{8{data_in[102][5]}} & 8'hfd)^
({{8{data_in[102][6]}} & 8'hd1)^
({{8{data_in[102][7]}} & 8'h89)^
({{8{data_in[103][0]}} & 8'h5a)^
({{8{data_in[103][1]}} & 8'hb4)^
({{8{data_in[103][2]}} & 8'h43)^
({{8{data_in[103][3]}} & 8'h86)^
({{8{data_in[103][4]}} & 8'h27)^
({{8{data_in[103][5]}} & 8'h4e)^
({{8{data_in[103][6]}} & 8'h9c)^
({{8{data_in[103][7]}} & 8'h13)^
({{8{data_in[104][0]}} & 8'h4)^
({{8{data_in[104][1]}} & 8'h8)^
({{8{data_in[104][2]}} & 8'h10)^
({{8{data_in[104][3]}} & 8'h20)^
({{8{data_in[104][4]}} & 8'h40)^
({{8{data_in[104][5]}} & 8'h80)^
({{8{data_in[104][6]}} & 8'h2b)^
({{8{data_in[104][7]}} & 8'h56)^
({{8{data_in[105][0]}} & 8'hbf)^
({{8{data_in[105][1]}} & 8'h55)^
({{8{data_in[105][2]}} & 8'haa)^
({{8{data_in[105][3]}} & 8'h7f)^
({{8{data_in[105][4]}} & 8'hfe)^
({{8{data_in[105][5]}} & 8'hd7)^
({{8{data_in[105][6]}} & 8'h85)^
({{8{data_in[105][7]}} & 8'h21)^
({{8{data_in[106][0]}} & 8'h96)^
({{8{data_in[106][1]}} & 8'h7)^
({{8{data_in[106][2]}} & 8'he)^
({{8{data_in[106][3]}} & 8'h1c)^
({{8{data_in[106][4]}} & 8'h38)^
({{8{data_in[106][5]}} & 8'h70)^
({{8{data_in[106][6]}} & 8'he0)^
({{8{data_in[106][7]}} & 8'heb)^
({{8{data_in[107][0]}} & 8'h92)^
({{8{data_in[107][1]}} & 8'hf)^
({{8{data_in[107][2]}} & 8'h1e)^
({{8{data_in[107][3]}} & 8'h3c)^
({{8{data_in[107][4]}} & 8'h78)^
({{8{data_in[107][5]}} & 8'hf0)^
({{8{data_in[107][6]}} & 8'hcb)^
({{8{data_in[107][7]}} & 8'hbd)^
({{8{data_in[108][0]}} & 8'hc5)^
({{8{data_in[108][1]}} & 8'ha1)^
({{8{data_in[108][2]}} & 8'h69)^
({{8{data_in[108][3]}} & 8'hd2)^
({{8{data_in[108][4]}} & 8'h8f)^
({{8{data_in[108][5]}} & 8'h35)^
({{8{data_in[108][6]}} & 8'h6a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[108][7]}} & 8'hd4)^
({{8{data_in[109][0]}} & 8'h74)^
({{8{data_in[109][1]}} & 8'he8)^
({{8{data_in[109][2]}} & 8'hfb)^
({{8{data_in[109][3]}} & 8'hdd)^
({{8{data_in[109][4]}} & 8'h91)^
({{8{data_in[109][5]}} & 8'h9)^
({{8{data_in[109][6]}} & 8'h12)^
({{8{data_in[109][7]}} & 8'h24)^
({{8{data_in[110][0]}} & 8'he0)^
({{8{data_in[110][1]}} & 8'heb)^
({{8{data_in[110][2]}} & 8'hfd)^
({{8{data_in[110][3]}} & 8'hd1)^
({{8{data_in[110][4]}} & 8'h89)^
({{8{data_in[110][5]}} & 8'h39)^
({{8{data_in[110][6]}} & 8'h72)^
({{8{data_in[110][7]}} & 8'he4)^
({{8{data_in[111][0]}} & 8'h15)^
({{8{data_in[111][1]}} & 8'h2a)^
({{8{data_in[111][2]}} & 8'h54)^
({{8{data_in[111][3]}} & 8'ha8)^
({{8{data_in[111][4]}} & 8'h7b)^
({{8{data_in[111][5]}} & 8'hf6)^
({{8{data_in[111][6]}} & 8'hc7)^
({{8{data_in[111][7]}} & 8'ha5)^
({{8{data_in[112][0]}} & 8'h34)^
({{8{data_in[112][1]}} & 8'h68)^
({{8{data_in[112][2]}} & 8'hd0)^
({{8{data_in[112][3]}} & 8'h8b)^
({{8{data_in[112][4]}} & 8'h3d)^
({{8{data_in[112][5]}} & 8'h7a)^
({{8{data_in[112][6]}} & 8'hf4)^
({{8{data_in[112][7]}} & 8'hc3)^
({{8{data_in[113][0]}} & 8'h31)^
({{8{data_in[113][1]}} & 8'h62)^
({{8{data_in[113][2]}} & 8'hc4)^
({{8{data_in[113][3]}} & 8'ha3)^
({{8{data_in[113][4]}} & 8'h6d)^
({{8{data_in[113][5]}} & 8'hda)^
({{8{data_in[113][6]}} & 8'h9f)^
({{8{data_in[113][7]}} & 8'h15)^
({{8{data_in[114][0]}} & 8'hbd)^
({{8{data_in[114][1]}} & 8'h51)^
({{8{data_in[114][2]}} & 8'ha2)^
({{8{data_in[114][3]}} & 8'h6f)^
({{8{data_in[114][4]}} & 8'hde)^
({{8{data_in[114][5]}} & 8'h97)^
({{8{data_in[114][6]}} & 8'h5)^
({{8{data_in[114][7]}} & 8'ha)^
({{8{data_in[115][0]}} & 8'h6a)^
({{8{data_in[115][1]}} & 8'hd4)^
({{8{data_in[115][2]}} & 8'h83)^
({{8{data_in[115][3]}} & 8'h2d)^
({{8{data_in[115][4]}} & 8'h5a)^
({{8{data_in[115][5]}} & 8'hb4)^
({{8{data_in[115][6]}} & 8'h43)^
({{8{data_in[115][7]}} & 8'h86)^
({{8{data_in[116][0]}} & 8'hef)^
({{8{data_in[116][1]}} & 8'hf5)^
({{8{data_in[116][2]}} & 8'hc1)^
({{8{data_in[116][3]}} & 8'ha9)^
({{8{data_in[116][4]}} & 8'h79)^
({{8{data_in[116][5]}} & 8'hf2)^
({{8{data_in[116][6]}} & 8'hcf)^
({{8{data_in[116][7]}} & 8'hb5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[117][0]}} & 8'he0)^
({{8{data_in[117][1]}} & 8'heb)^
({{8{data_in[117][2]}} & 8'hfd)^
({{8{data_in[117][3]}} & 8'hd1)^
({{8{data_in[117][4]}} & 8'h89)^
({{8{data_in[117][5]}} & 8'h39)^
({{8{data_in[117][6]}} & 8'h72)^
({{8{data_in[117][7]}} & 8'he4)^
({{8{data_in[118][0]}} & 8'h62)^
({{8{data_in[118][1]}} & 8'hc4)^
({{8{data_in[118][2]}} & 8'ha3)^
({{8{data_in[118][3]}} & 8'h6d)^
({{8{data_in[118][4]}} & 8'hda)^
({{8{data_in[118][5]}} & 8'h9f)^
({{8{data_in[118][6]}} & 8'h15)^
({{8{data_in[118][7]}} & 8'h2a)^
({{8{data_in[119][0]}} & 8'h69)^
({{8{data_in[119][1]}} & 8'hd2)^
({{8{data_in[119][2]}} & 8'h8f)^
({{8{data_in[119][3]}} & 8'h35)^
({{8{data_in[119][4]}} & 8'h6a)^
({{8{data_in[119][5]}} & 8'hd4)^
({{8{data_in[119][6]}} & 8'h83)^
({{8{data_in[119][7]}} & 8'h2d)^
({{8{data_in[120][0]}} & 8'h52)^
({{8{data_in[120][1]}} & 8'ha4)^
({{8{data_in[120][2]}} & 8'h63)^
({{8{data_in[120][3]}} & 8'hc6)^
({{8{data_in[120][4]}} & 8'ha7)^
({{8{data_in[120][5]}} & 8'h65)^
({{8{data_in[120][6]}} & 8'hca)^
({{8{data_in[120][7]}} & 8'hbf)^
({{8{data_in[121][0]}} & 8'hd3)^
({{8{data_in[121][1]}} & 8'h8d)^
({{8{data_in[121][2]}} & 8'h31)^
({{8{data_in[121][3]}} & 8'h62)^
({{8{data_in[121][4]}} & 8'hc4)^
({{8{data_in[121][5]}} & 8'ha3)^
({{8{data_in[121][6]}} & 8'h6d)^
({{8{data_in[121][7]}} & 8'hda)^
({{8{data_in[122][0]}} & 8'h41)^
({{8{data_in[122][1]}} & 8'h82)^
({{8{data_in[122][2]}} & 8'h2f)^
({{8{data_in[122][3]}} & 8'h5e)^
({{8{data_in[122][4]}} & 8'hbc)^
({{8{data_in[122][5]}} & 8'h53)^
({{8{data_in[122][6]}} & 8'ha6)^
({{8{data_in[122][7]}} & 8'h67)^
({{8{data_in[123][0]}} & 8'hb0)^
({{8{data_in[123][1]}} & 8'h4b)^
({{8{data_in[123][2]}} & 8'h96)^
({{8{data_in[123][3]}} & 8'h7)^
({{8{data_in[123][4]}} & 8'he)^
({{8{data_in[123][5]}} & 8'h1c)^
({{8{data_in[123][6]}} & 8'h38)^
({{8{data_in[123][7]}} & 8'h70)^
({{8{data_in[124][0]}} & 8'h6f)^
({{8{data_in[124][1]}} & 8'hde)^
({{8{data_in[124][2]}} & 8'h97)^
({{8{data_in[124][3]}} & 8'h5)^
({{8{data_in[124][4]}} & 8'ha)^
({{8{data_in[124][5]}} & 8'h14)^
({{8{data_in[124][6]}} & 8'h28)^
({{8{data_in[124][7]}} & 8'h50)^
({{8{data_in[125][0]}} & 8'hd7)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[125][1]}} & 8'h85)^
({{8{data_in[125][2]}} & 8'h21)^
({{8{data_in[125][3]}} & 8'h42)^
({{8{data_in[125][4]}} & 8'h84)^
({{8{data_in[125][5]}} & 8'h23)^
({{8{data_in[125][6]}} & 8'h46)^
({{8{data_in[125][7]}} & 8'h8c)^
({{8{data_in[126][0]}} & 8'he)^
({{8{data_in[126][1]}} & 8'h1c)^
({{8{data_in[126][2]}} & 8'h38)^
({{8{data_in[126][3]}} & 8'h70)^
({{8{data_in[126][4]}} & 8'he0)^
({{8{data_in[126][5]}} & 8'heb)^
({{8{data_in[126][6]}} & 8'hfd)^
({{8{data_in[126][7]}} & 8'hd1)^
({{8{data_in[127][0]}} & 8'haa)^
({{8{data_in[127][1]}} & 8'h7f)^
({{8{data_in[127][2]}} & 8'hfe)^
({{8{data_in[127][3]}} & 8'hd7)^
({{8{data_in[127][4]}} & 8'h85)^
({{8{data_in[127][5]}} & 8'h21)^
({{8{data_in[127][6]}} & 8'h42)^
({{8{data_in[127][7]}} & 8'h84)^
({{8{data_in[128][0]}} & 8'ha9)^
({{8{data_in[128][1]}} & 8'h79)^
({{8{data_in[128][2]}} & 8'hf2)^
({{8{data_in[128][3]}} & 8'hcf)^
({{8{data_in[128][4]}} & 8'hb5)^
({{8{data_in[128][5]}} & 8'h41)^
({{8{data_in[128][6]}} & 8'h82)^
({{8{data_in[128][7]}} & 8'h2f)^
({{8{data_in[129][0]}} & 8'h83)^
({{8{data_in[129][1]}} & 8'h2d)^
({{8{data_in[129][2]}} & 8'h5a)^
({{8{data_in[129][3]}} & 8'hb4)^
({{8{data_in[129][4]}} & 8'h43)^
({{8{data_in[129][5]}} & 8'h86)^
({{8{data_in[129][6]}} & 8'h27)^
({{8{data_in[129][7]}} & 8'h4e)^
({{8{data_in[130][0]}} & 8'hf1)^
({{8{data_in[130][1]}} & 8'hc9)^
({{8{data_in[130][2]}} & 8'hb9)^
({{8{data_in[130][3]}} & 8'h59)^
({{8{data_in[130][4]}} & 8'hb2)^
({{8{data_in[130][5]}} & 8'h4f)^
({{8{data_in[130][6]}} & 8'h9e)^
({{8{data_in[130][7]}} & 8'h17)^
({{8{data_in[131][0]}} & 8'ha4)^
({{8{data_in[131][1]}} & 8'h63)^
({{8{data_in[131][2]}} & 8'hc6)^
({{8{data_in[131][3]}} & 8'ha7)^
({{8{data_in[131][4]}} & 8'h65)^
({{8{data_in[131][5]}} & 8'hca)^
({{8{data_in[131][6]}} & 8'hbf)^
({{8{data_in[131][7]}} & 8'h55)^
({{8{data_in[132][0]}} & 8'h12)^
({{8{data_in[132][1]}} & 8'h24)^
({{8{data_in[132][2]}} & 8'h48)^
({{8{data_in[132][3]}} & 8'h90)^
({{8{data_in[132][4]}} & 8'hb)^
({{8{data_in[132][5]}} & 8'h16)^
({{8{data_in[132][6]}} & 8'h2c)^
({{8{data_in[132][7]}} & 8'h58)^
({{8{data_in[133][0]}} & 8'he8)^
({{8{data_in[133][1]}} & 8'hfb)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[133][2]}} & 8'hdd)^
({{8{data_in[133][3]}} & 8'h91)^
({{8{data_in[133][4]}} & 8'h9)^
({{8{data_in[133][5]}} & 8'h12)^
({{8{data_in[133][6]}} & 8'h24)^
({{8{data_in[133][7]}} & 8'h48)^
({{8{data_in[134][0]}} & 8'h49)^
({{8{data_in[134][1]}} & 8'h92)^
({{8{data_in[134][2]}} & 8'hf)^
({{8{data_in[134][3]}} & 8'h1e)^
({{8{data_in[134][4]}} & 8'h3c)^
({{8{data_in[134][5]}} & 8'h78)^
({{8{data_in[134][6]}} & 8'hf0)^
({{8{data_in[134][7]}} & 8'hcb)^
({{8{data_in[135][0]}} & 8'h11)^
({{8{data_in[135][1]}} & 8'h22)^
({{8{data_in[135][2]}} & 8'h44)^
({{8{data_in[135][3]}} & 8'h88)^
({{8{data_in[135][4]}} & 8'h3b)^
({{8{data_in[135][5]}} & 8'h76)^
({{8{data_in[135][6]}} & 8'hec)^
({{8{data_in[135][7]}} & 8'hf3)^
({{8{data_in[136][0]}} & 8'h3b)^
({{8{data_in[136][1]}} & 8'h76)^
({{8{data_in[136][2]}} & 8'hec)^
({{8{data_in[136][3]}} & 8'hf3)^
({{8{data_in[136][4]}} & 8'hcd)^
({{8{data_in[136][5]}} & 8'hb1)^
({{8{data_in[136][6]}} & 8'h49)^
({{8{data_in[136][7]}} & 8'h92)^
({{8{data_in[137][0]}} & 8'h5f)^
({{8{data_in[137][1]}} & 8'hbe)^
({{8{data_in[137][2]}} & 8'h57)^
({{8{data_in[137][3]}} & 8'hae)^
({{8{data_in[137][4]}} & 8'h77)^
({{8{data_in[137][5]}} & 8'hee)^
({{8{data_in[137][6]}} & 8'hf7)^
({{8{data_in[137][7]}} & 8'hc5)^
({{8{data_in[138][0]}} & 8'h10)^
({{8{data_in[138][1]}} & 8'h20)^
({{8{data_in[138][2]}} & 8'h40)^
({{8{data_in[138][3]}} & 8'h80)^
({{8{data_in[138][4]}} & 8'h2b)^
({{8{data_in[138][5]}} & 8'h56)^
({{8{data_in[138][6]}} & 8'hac)^
({{8{data_in[138][7]}} & 8'h73)^
({{8{data_in[139][0]}} & 8'he7)^
({{8{data_in[139][1]}} & 8'he5)^
({{8{data_in[139][2]}} & 8'he1)^
({{8{data_in[139][3]}} & 8'he9)^
({{8{data_in[139][4]}} & 8'hf9)^
({{8{data_in[139][5]}} & 8'hd9)^
({{8{data_in[139][6]}} & 8'h99)^
({{8{data_in[139][7]}} & 8'h19)^
({{8{data_in[140][0]}} & 8'h99)^
({{8{data_in[140][1]}} & 8'h19)^
({{8{data_in[140][2]}} & 8'h32)^
({{8{data_in[140][3]}} & 8'h64)^
({{8{data_in[140][4]}} & 8'hc8)^
({{8{data_in[140][5]}} & 8'hbb)^
({{8{data_in[140][6]}} & 8'h5d)^
({{8{data_in[140][7]}} & 8'hba)^
({{8{data_in[141][0]}} & 8'hd6)^
({{8{data_in[141][1]}} & 8'h87)^
({{8{data_in[141][2]}} & 8'h25)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[141][3]}} & 8'h4a)^
({{8{data_in[141][4]}} & 8'h94)^
({{8{data_in[141][5]}} & 8'h3)^
({{8{data_in[141][6]}} & 8'h6)^
({{8{data_in[141][7]}} & 8'hc)^
({{8{data_in[142][0]}} & 8'hd5)^
({{8{data_in[142][1]}} & 8'h81)^
({{8{data_in[142][2]}} & 8'h29)^
({{8{data_in[142][3]}} & 8'h52)^
({{8{data_in[142][4]}} & 8'ha4)^
({{8{data_in[142][5]}} & 8'h63)^
({{8{data_in[142][6]}} & 8'hc6)^
({{8{data_in[142][7]}} & 8'ha7)^
({{8{data_in[143][0]}} & 8'hff)^
({{8{data_in[143][1]}} & 8'hd5)^
({{8{data_in[143][2]}} & 8'h81)^
({{8{data_in[143][3]}} & 8'h29)^
({{8{data_in[143][4]}} & 8'h52)^
({{8{data_in[143][5]}} & 8'ha4)^
({{8{data_in[143][6]}} & 8'h63)^
({{8{data_in[143][7]}} & 8'hc6)^
({{8{data_in[144][0]}} & 8'hf7)^
({{8{data_in[144][1]}} & 8'hc5)^
({{8{data_in[144][2]}} & 8'ha1)^
({{8{data_in[144][3]}} & 8'h69)^
({{8{data_in[144][4]}} & 8'hd2)^
({{8{data_in[144][5]}} & 8'h8f)^
({{8{data_in[144][6]}} & 8'h35)^
({{8{data_in[144][7]}} & 8'h6a)^
({{8{data_in[145][0]}} & 8'h28)^
({{8{data_in[145][1]}} & 8'h50)^
({{8{data_in[145][2]}} & 8'ha0)^
({{8{data_in[145][3]}} & 8'h6b)^
({{8{data_in[145][4]}} & 8'hd6)^
({{8{data_in[145][5]}} & 8'h87)^
({{8{data_in[145][6]}} & 8'h25)^
({{8{data_in[145][7]}} & 8'h4a)^
({{8{data_in[146][0]}} & 8'hf1)^
({{8{data_in[146][1]}} & 8'hc9)^
({{8{data_in[146][2]}} & 8'hb9)^
({{8{data_in[146][3]}} & 8'h59)^
({{8{data_in[146][4]}} & 8'hb2)^
({{8{data_in[146][5]}} & 8'h4f)^
({{8{data_in[146][6]}} & 8'h9e)^
({{8{data_in[146][7]}} & 8'h17)^
({{8{data_in[147][0]}} & 8'hcc)^
({{8{data_in[147][1]}} & 8'hb3)^
({{8{data_in[147][2]}} & 8'h4d)^
({{8{data_in[147][3]}} & 8'h9a)^
({{8{data_in[147][4]}} & 8'h1f)^
({{8{data_in[147][5]}} & 8'h3e)^
({{8{data_in[147][6]}} & 8'h7c)^
({{8{data_in[147][7]}} & 8'hf8)^
({{8{data_in[148][0]}} & 8'h31)^
({{8{data_in[148][1]}} & 8'h62)^
({{8{data_in[148][2]}} & 8'hc4)^
({{8{data_in[148][3]}} & 8'ha3)^
({{8{data_in[148][4]}} & 8'h6d)^
({{8{data_in[148][5]}} & 8'hda)^
({{8{data_in[148][6]}} & 8'h9f)^
({{8{data_in[148][7]}} & 8'h15)^
({{8{data_in[149][0]}} & 8'h40)^
({{8{data_in[149][1]}} & 8'h80)^
({{8{data_in[149][2]}} & 8'h2b)^
({{8{data_in[149][3]}} & 8'h56)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[149][4]}} & 8'hac)^
({{8{data_in[149][5]}} & 8'h73)^
({{8{data_in[149][6]}} & 8'he6)^
({{8{data_in[149][7]}} & 8'he7)^
({{8{data_in[150][0]}} & 8'h98)^
({{8{data_in[150][1]}} & 8'h1b)^
({{8{data_in[150][2]}} & 8'h36)^
({{8{data_in[150][3]}} & 8'h6c)^
({{8{data_in[150][4]}} & 8'hd8)^
({{8{data_in[150][5]}} & 8'h9b)^
({{8{data_in[150][6]}} & 8'h1d)^
({{8{data_in[150][7]}} & 8'h3a)^
({{8{data_in[151][0]}} & 8'h16)^
({{8{data_in[151][1]}} & 8'h2c)^
({{8{data_in[151][2]}} & 8'h58)^
({{8{data_in[151][3]}} & 8'hb0)^
({{8{data_in[151][4]}} & 8'h4b)^
({{8{data_in[151][5]}} & 8'h96)^
({{8{data_in[151][6]}} & 8'h7)^
({{8{data_in[151][7]}} & 8'he)^
({{8{data_in[152][0]}} & 8'h8d)^
({{8{data_in[152][1]}} & 8'h31)^
({{8{data_in[152][2]}} & 8'h62)^
({{8{data_in[152][3]}} & 8'hc4)^
({{8{data_in[152][4]}} & 8'ha3)^
({{8{data_in[152][5]}} & 8'h6d)^
({{8{data_in[152][6]}} & 8'hda)^
({{8{data_in[152][7]}} & 8'h9f)^
({{8{data_in[153][0]}} & 8'h6)^
({{8{data_in[153][1]}} & 8'hc)^
({{8{data_in[153][2]}} & 8'h18)^
({{8{data_in[153][3]}} & 8'h30)^
({{8{data_in[153][4]}} & 8'h60)^
({{8{data_in[153][5]}} & 8'hc0)^
({{8{data_in[153][6]}} & 8'hab)^
({{8{data_in[153][7]}} & 8'h7d)^
({{8{data_in[154][0]}} & 8'hdb)^
({{8{data_in[154][1]}} & 8'h9d)^
({{8{data_in[154][2]}} & 8'h11)^
({{8{data_in[154][3]}} & 8'h22)^
({{8{data_in[154][4]}} & 8'h44)^
({{8{data_in[154][5]}} & 8'h88)^
({{8{data_in[154][6]}} & 8'h3b)^
({{8{data_in[154][7]}} & 8'h76)^
({{8{data_in[155][0]}} & 8'h43)^
({{8{data_in[155][1]}} & 8'h86)^
({{8{data_in[155][2]}} & 8'h27)^
({{8{data_in[155][3]}} & 8'h4e)^
({{8{data_in[155][4]}} & 8'h9c)^
({{8{data_in[155][5]}} & 8'h13)^
({{8{data_in[155][6]}} & 8'h26)^
({{8{data_in[155][7]}} & 8'h4c)^
({{8{data_in[156][0]}} & 8'ha3)^
({{8{data_in[156][1]}} & 8'h6d)^
({{8{data_in[156][2]}} & 8'hda)^
({{8{data_in[156][3]}} & 8'h9f)^
({{8{data_in[156][4]}} & 8'h15)^
({{8{data_in[156][5]}} & 8'h2a)^
({{8{data_in[156][6]}} & 8'h54)^
({{8{data_in[156][7]}} & 8'ha8)^
({{8{data_in[157][0]}} & 8'h6b)^
({{8{data_in[157][1]}} & 8'hd6)^
({{8{data_in[157][2]}} & 8'h87)^
({{8{data_in[157][3]}} & 8'h25)^
({{8{data_in[157][4]}} & 8'h4a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[157][5]}} & 8'h94)^
({{8{data_in[157][6]}} & 8'h3)^
({{8{data_in[157][7]}} & 8'h6)^
({{8{data_in[158][0]}} & 8'hf)^
({{8{data_in[158][1]}} & 8'h1e)^
({{8{data_in[158][2]}} & 8'h3c)^
({{8{data_in[158][3]}} & 8'h78)^
({{8{data_in[158][4]}} & 8'hf0)^
({{8{data_in[158][5]}} & 8'hcb)^
({{8{data_in[158][6]}} & 8'hbd)^
({{8{data_in[158][7]}} & 8'h51)^
({{8{data_in[159][0]}} & 8'h83)^
({{8{data_in[159][1]}} & 8'h2d)^
({{8{data_in[159][2]}} & 8'h5a)^
({{8{data_in[159][3]}} & 8'hb4)^
({{8{data_in[159][4]}} & 8'h43)^
({{8{data_in[159][5]}} & 8'h86)^
({{8{data_in[159][6]}} & 8'h27)^
({{8{data_in[159][7]}} & 8'h4e)^
({{8{data_in[160][0]}} & 8'h59)^
({{8{data_in[160][1]}} & 8'hb2)^
({{8{data_in[160][2]}} & 8'h4f)^
({{8{data_in[160][3]}} & 8'h9e)^
({{8{data_in[160][4]}} & 8'h17)^
({{8{data_in[160][5]}} & 8'h2e)^
({{8{data_in[160][6]}} & 8'h5c)^
({{8{data_in[160][7]}} & 8'hb8)^
({{8{data_in[161][0]}} & 8'hde)^
({{8{data_in[161][1]}} & 8'h97)^
({{8{data_in[161][2]}} & 8'h5)^
({{8{data_in[161][3]}} & 8'ha)^
({{8{data_in[161][4]}} & 8'h14)^
({{8{data_in[161][5]}} & 8'h28)^
({{8{data_in[161][6]}} & 8'h50)^
({{8{data_in[161][7]}} & 8'ha0)^
({{8{data_in[162][0]}} & 8'hc1)^
({{8{data_in[162][1]}} & 8'ha9)^
({{8{data_in[162][2]}} & 8'h79)^
({{8{data_in[162][3]}} & 8'hf2)^
({{8{data_in[162][4]}} & 8'hcf)^
({{8{data_in[162][5]}} & 8'hb5)^
({{8{data_in[162][6]}} & 8'h41)^
({{8{data_in[162][7]}} & 8'h82)^
({{8{data_in[163][0]}} & 8'hc5)^
({{8{data_in[163][1]}} & 8'ha1)^
({{8{data_in[163][2]}} & 8'h69)^
({{8{data_in[163][3]}} & 8'hd2)^
({{8{data_in[163][4]}} & 8'h8f)^
({{8{data_in[163][5]}} & 8'h35)^
({{8{data_in[163][6]}} & 8'h6a)^
({{8{data_in[163][7]}} & 8'hd4)^
({{8{data_in[164][0]}} & 8'h45)^
({{8{data_in[164][1]}} & 8'h8a)^
({{8{data_in[164][2]}} & 8'h3f)^
({{8{data_in[164][3]}} & 8'h7e)^
({{8{data_in[164][4]}} & 8'hfc)^
({{8{data_in[164][5]}} & 8'hd3)^
({{8{data_in[164][6]}} & 8'h8d)^
({{8{data_in[164][7]}} & 8'h31)^
({{8{data_in[165][0]}} & 8'h1b)^
({{8{data_in[165][1]}} & 8'h36)^
({{8{data_in[165][2]}} & 8'h6c)^
({{8{data_in[165][3]}} & 8'hd8)^
({{8{data_in[165][4]}} & 8'h9b)^
({{8{data_in[165][5]}} & 8'h1d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[165][6]}} & 8'h3a)^
({{8{data_in[165][7]}} & 8'h74)^
({{8{data_in[166][0]}} & 8'h97)^
({{8{data_in[166][1]}} & 8'h5)^
({{8{data_in[166][2]}} & 8'ha)^
({{8{data_in[166][3]}} & 8'h14)^
({{8{data_in[166][4]}} & 8'h28)^
({{8{data_in[166][5]}} & 8'h50)^
({{8{data_in[166][6]}} & 8'ha0)^
({{8{data_in[166][7]}} & 8'h6b)^
({{8{data_in[167][0]}} & 8'hb1)^
({{8{data_in[167][1]}} & 8'h49)^
({{8{data_in[167][2]}} & 8'h92)^
({{8{data_in[167][3]}} & 8'hf)^
({{8{data_in[167][4]}} & 8'h1e)^
({{8{data_in[167][5]}} & 8'h3c)^
({{8{data_in[167][6]}} & 8'h78)^
({{8{data_in[167][7]}} & 8'hf0)^
({{8{data_in[168][0]}} & 8'h15)^
({{8{data_in[168][1]}} & 8'h2a)^
({{8{data_in[168][2]}} & 8'h54)^
({{8{data_in[168][3]}} & 8'ha8)^
({{8{data_in[168][4]}} & 8'h7b)^
({{8{data_in[168][5]}} & 8'hf6)^
({{8{data_in[168][6]}} & 8'hc7)^
({{8{data_in[168][7]}} & 8'ha5)^
({{8{data_in[169][0]}} & 8'hf0)^
({{8{data_in[169][1]}} & 8'hcb)^
({{8{data_in[169][2]}} & 8'hbd)^
({{8{data_in[169][3]}} & 8'h51)^
({{8{data_in[169][4]}} & 8'ha2)^
({{8{data_in[169][5]}} & 8'h6f)^
({{8{data_in[169][6]}} & 8'hde)^
({{8{data_in[169][7]}} & 8'h97)^
({{8{data_in[170][0]}} & 8'h4f)^
({{8{data_in[170][1]}} & 8'h9e)^
({{8{data_in[170][2]}} & 8'h17)^
({{8{data_in[170][3]}} & 8'h2e)^
({{8{data_in[170][4]}} & 8'h5c)^
({{8{data_in[170][5]}} & 8'hb8)^
({{8{data_in[170][6]}} & 8'h5b)^
({{8{data_in[170][7]}} & 8'hb6)^
({{8{data_in[171][0]}} & 8'hde)^
({{8{data_in[171][1]}} & 8'h97)^
({{8{data_in[171][2]}} & 8'h5)^
({{8{data_in[171][3]}} & 8'ha)^
({{8{data_in[171][4]}} & 8'h14)^
({{8{data_in[171][5]}} & 8'h28)^
({{8{data_in[171][6]}} & 8'h50)^
({{8{data_in[171][7]}} & 8'ha0)^
({{8{data_in[172][0]}} & 8'h9f)^
({{8{data_in[172][1]}} & 8'h15)^
({{8{data_in[172][2]}} & 8'h2a)^
({{8{data_in[172][3]}} & 8'h54)^
({{8{data_in[172][4]}} & 8'ha8)^
({{8{data_in[172][5]}} & 8'h7b)^
({{8{data_in[172][6]}} & 8'hf6)^
({{8{data_in[172][7]}} & 8'hc7)^
({{8{data_in[173][0]}} & 8'hbb)^
({{8{data_in[173][1]}} & 8'h5d)^
({{8{data_in[173][2]}} & 8'hba)^
({{8{data_in[173][3]}} & 8'h5f)^
({{8{data_in[173][4]}} & 8'hbe)^
({{8{data_in[173][5]}} & 8'h57)^
({{8{data_in[173][6]}} & 8'hae)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[173][7]}} & 8'h77)^
({{8{data_in[174][0]}} & 8'hbb)^
({{8{data_in[174][1]}} & 8'h5d)^
({{8{data_in[174][2]}} & 8'hba)^
({{8{data_in[174][3]}} & 8'h5f)^
({{8{data_in[174][4]}} & 8'hbe)^
({{8{data_in[174][5]}} & 8'h57)^
({{8{data_in[174][6]}} & 8'hae)^
({{8{data_in[174][7]}} & 8'h77)^
({{8{data_in[175][0]}} & 8'hf7)^
({{8{data_in[175][1]}} & 8'hc5)^
({{8{data_in[175][2]}} & 8'ha1)^
({{8{data_in[175][3]}} & 8'h69)^
({{8{data_in[175][4]}} & 8'hd2)^
({{8{data_in[175][5]}} & 8'h8f)^
({{8{data_in[175][6]}} & 8'h35)^
({{8{data_in[175][7]}} & 8'h6a)^
({{8{data_in[176][0]}} & 8'hd2)^
({{8{data_in[176][1]}} & 8'h8f)^
({{8{data_in[176][2]}} & 8'h35)^
({{8{data_in[176][3]}} & 8'h6a)^
({{8{data_in[176][4]}} & 8'hd4)^
({{8{data_in[176][5]}} & 8'h83)^
({{8{data_in[176][6]}} & 8'h2d)^
({{8{data_in[176][7]}} & 8'h5a)^
({{8{data_in[177][0]}} & 8'h50)^
({{8{data_in[177][1]}} & 8'ha0)^
({{8{data_in[177][2]}} & 8'h6b)^
({{8{data_in[177][3]}} & 8'hd6)^
({{8{data_in[177][4]}} & 8'h87)^
({{8{data_in[177][5]}} & 8'h25)^
({{8{data_in[177][6]}} & 8'h4a)^
({{8{data_in[177][7]}} & 8'h94)^
({{8{data_in[178][0]}} & 8'h86)^
({{8{data_in[178][1]}} & 8'h27)^
({{8{data_in[178][2]}} & 8'h4e)^
({{8{data_in[178][3]}} & 8'h9c)^
({{8{data_in[178][4]}} & 8'h13)^
({{8{data_in[178][5]}} & 8'h26)^
({{8{data_in[178][6]}} & 8'h4c)^
({{8{data_in[178][7]}} & 8'h98)^
({{8{data_in[179][0]}} & 8'h55)^
({{8{data_in[179][1]}} & 8'haa)^
({{8{data_in[179][2]}} & 8'h7f)^
({{8{data_in[179][3]}} & 8'hfe)^
({{8{data_in[179][4]}} & 8'hd7)^
({{8{data_in[179][5]}} & 8'h85)^
({{8{data_in[179][6]}} & 8'h21)^
({{8{data_in[179][7]}} & 8'h42)^
({{8{data_in[180][0]}} & 8'hf1)^
({{8{data_in[180][1]}} & 8'hc9)^
({{8{data_in[180][2]}} & 8'hb9)^
({{8{data_in[180][3]}} & 8'h59)^
({{8{data_in[180][4]}} & 8'hb2)^
({{8{data_in[180][5]}} & 8'h4f)^
({{8{data_in[180][6]}} & 8'h9e)^
({{8{data_in[180][7]}} & 8'h17)^
({{8{data_in[181][0]}} & 8'hc6)^
({{8{data_in[181][1]}} & 8'ha7)^
({{8{data_in[181][2]}} & 8'h65)^
({{8{data_in[181][3]}} & 8'hca)^
({{8{data_in[181][4]}} & 8'hbf)^
({{8{data_in[181][5]}} & 8'h55)^
({{8{data_in[181][6]}} & 8'haa)^
({{8{data_in[181][7]}} & 8'h7f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[182][0]}} & 8'h85)^
({{8{data_in[182][1]}} & 8'h21)^
({{8{data_in[182][2]}} & 8'h42)^
({{8{data_in[182][3]}} & 8'h84)^
({{8{data_in[182][4]}} & 8'h23)^
({{8{data_in[182][5]}} & 8'h46)^
({{8{data_in[182][6]}} & 8'h8c)^
({{8{data_in[182][7]}} & 8'h33)^
({{8{data_in[183][0]}} & 8'h5)^
({{8{data_in[183][1]}} & 8'ha)^
({{8{data_in[183][2]}} & 8'h14)^
({{8{data_in[183][3]}} & 8'h28)^
({{8{data_in[183][4]}} & 8'h50)^
({{8{data_in[183][5]}} & 8'ha0)^
({{8{data_in[183][6]}} & 8'h6b)^
({{8{data_in[183][7]}} & 8'hd6)^
({{8{data_in[184][0]}} & 8'h82)^
({{8{data_in[184][1]}} & 8'h2f)^
({{8{data_in[184][2]}} & 8'h5e)^
({{8{data_in[184][3]}} & 8'hbc)^
({{8{data_in[184][4]}} & 8'h53)^
({{8{data_in[184][5]}} & 8'ha6)^
({{8{data_in[184][6]}} & 8'h67)^
({{8{data_in[184][7]}} & 8'hce)^
({{8{data_in[185][0]}} & 8'h4c)^
({{8{data_in[185][1]}} & 8'h98)^
({{8{data_in[185][2]}} & 8'h1b)^
({{8{data_in[185][3]}} & 8'h36)^
({{8{data_in[185][4]}} & 8'h6c)^
({{8{data_in[185][5]}} & 8'hd8)^
({{8{data_in[185][6]}} & 8'h9b)^
({{8{data_in[185][7]}} & 8'h1d)^
({{8{data_in[186][0]}} & 8'hb4)^
({{8{data_in[186][1]}} & 8'h43)^
({{8{data_in[186][2]}} & 8'h86)^
({{8{data_in[186][3]}} & 8'h27)^
({{8{data_in[186][4]}} & 8'h4e)^
({{8{data_in[186][5]}} & 8'h9c)^
({{8{data_in[186][6]}} & 8'h13)^
({{8{data_in[186][7]}} & 8'h26)^
({{8{data_in[187][0]}} & 8'h24)^
({{8{data_in[187][1]}} & 8'h48)^
({{8{data_in[187][2]}} & 8'h90)^
({{8{data_in[187][3]}} & 8'hb)^
({{8{data_in[187][4]}} & 8'h16)^
({{8{data_in[187][5]}} & 8'h2c)^
({{8{data_in[187][6]}} & 8'h58)^
({{8{data_in[187][7]}} & 8'hb0)^
({{8{data_in[188][0]}} & 8'h69)^
({{8{data_in[188][1]}} & 8'hd2)^
({{8{data_in[188][2]}} & 8'h8f)^
({{8{data_in[188][3]}} & 8'h35)^
({{8{data_in[188][4]}} & 8'h6a)^
({{8{data_in[188][5]}} & 8'hd4)^
({{8{data_in[188][6]}} & 8'h83)^
({{8{data_in[188][7]}} & 8'h2d)^
({{8{data_in[189][0]}} & 8'h24)^
({{8{data_in[189][1]}} & 8'h48)^
({{8{data_in[189][2]}} & 8'h90)^
({{8{data_in[189][3]}} & 8'hb)^
({{8{data_in[189][4]}} & 8'h16)^
({{8{data_in[189][5]}} & 8'h2c)^
({{8{data_in[189][6]}} & 8'h58)^
({{8{data_in[189][7]}} & 8'hb0)^
({{8{data_in[190][0]}} & 8'hae)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[190][1]}} & 8'h77)^
({{8{data_in[190][2]}} & 8'hee)^
({{8{data_in[190][3]}} & 8'hf7)^
({{8{data_in[190][4]}} & 8'hc5)^
({{8{data_in[190][5]}} & 8'ha1)^
({{8{data_in[190][6]}} & 8'h69)^
({{8{data_in[190][7]}} & 8'hd2)^
({{8{data_in[191][0]}} & 8'h30)^
({{8{data_in[191][1]}} & 8'h60)^
({{8{data_in[191][2]}} & 8'hc0)^
({{8{data_in[191][3]}} & 8'hab)^
({{8{data_in[191][4]}} & 8'h7d)^
({{8{data_in[191][5]}} & 8'hfa)^
({{8{data_in[191][6]}} & 8'hdf)^
({{8{data_in[191][7]}} & 8'h95)^
({{8{data_in[192][0]}} & 8'h95)^
({{8{data_in[192][1]}} & 8'h1)^
({{8{data_in[192][2]}} & 8'h2)^
({{8{data_in[192][3]}} & 8'h4)^
({{8{data_in[192][4]}} & 8'h8)^
({{8{data_in[192][5]}} & 8'h10)^
({{8{data_in[192][6]}} & 8'h20)^
({{8{data_in[192][7]}} & 8'h40)^
({{8{data_in[193][0]}} & 8'hed)^
({{8{data_in[193][1]}} & 8'hf1)^
({{8{data_in[193][2]}} & 8'hc9)^
({{8{data_in[193][3]}} & 8'hb9)^
({{8{data_in[193][4]}} & 8'h59)^
({{8{data_in[193][5]}} & 8'hb2)^
({{8{data_in[193][6]}} & 8'h4f)^
({{8{data_in[193][7]}} & 8'h9e)^
({{8{data_in[194][0]}} & 8'he1)^
({{8{data_in[194][1]}} & 8'he9)^
({{8{data_in[194][2]}} & 8'hf9)^
({{8{data_in[194][3]}} & 8'hd9)^
({{8{data_in[194][4]}} & 8'h99)^
({{8{data_in[194][5]}} & 8'h19)^
({{8{data_in[194][6]}} & 8'h32)^
({{8{data_in[194][7]}} & 8'h64)^
({{8{data_in[195][0]}} & 8'h2)^
({{8{data_in[195][1]}} & 8'h4)^
({{8{data_in[195][2]}} & 8'h8)^
({{8{data_in[195][3]}} & 8'h10)^
({{8{data_in[195][4]}} & 8'h20)^
({{8{data_in[195][5]}} & 8'h40)^
({{8{data_in[195][6]}} & 8'h80)^
({{8{data_in[195][7]}} & 8'h2b)^
({{8{data_in[196][0]}} & 8'hc8)^
({{8{data_in[196][1]}} & 8'hbb)^
({{8{data_in[196][2]}} & 8'h5d)^
({{8{data_in[196][3]}} & 8'hba)^
({{8{data_in[196][4]}} & 8'h5f)^
({{8{data_in[196][5]}} & 8'hbe)^
({{8{data_in[196][6]}} & 8'h57)^
({{8{data_in[196][7]}} & 8'hae)^
({{8{data_in[197][0]}} & 8'h8)^
({{8{data_in[197][1]}} & 8'h10)^
({{8{data_in[197][2]}} & 8'h20)^
({{8{data_in[197][3]}} & 8'h40)^
({{8{data_in[197][4]}} & 8'h80)^
({{8{data_in[197][5]}} & 8'h2b)^
({{8{data_in[197][6]}} & 8'h56)^
({{8{data_in[197][7]}} & 8'hac)^
({{8{data_in[198][0]}} & 8'hf5)^
({{8{data_in[198][1]}} & 8'hc1)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[198][2]}} & 8'ha9)^
({{8{data_in[198][3]}} & 8'h79)^
({{8{data_in[198][4]}} & 8'hf2)^
({{8{data_in[198][5]}} & 8'hcf)^
({{8{data_in[198][6]}} & 8'hb5)^
({{8{data_in[198][7]}} & 8'h41)^
({{8{data_in[199][0]}} & 8'hca)^
({{8{data_in[199][1]}} & 8'hbf)^
({{8{data_in[199][2]}} & 8'h55)^
({{8{data_in[199][3]}} & 8'haa)^
({{8{data_in[199][4]}} & 8'h7f)^
({{8{data_in[199][5]}} & 8'hfe)^
({{8{data_in[199][6]}} & 8'hd7)^
({{8{data_in[199][7]}} & 8'h85)^
({{8{data_in[200][0]}} & 8'h61)^
({{8{data_in[200][1]}} & 8'hc2)^
({{8{data_in[200][2]}} & 8'haf)^
({{8{data_in[200][3]}} & 8'h75)^
({{8{data_in[200][4]}} & 8'hea)^
({{8{data_in[200][5]}} & 8'hff)^
({{8{data_in[200][6]}} & 8'hd5)^
({{8{data_in[200][7]}} & 8'h81)^
({{8{data_in[201][0]}} & 8'h73)^
({{8{data_in[201][1]}} & 8'he6)^
({{8{data_in[201][2]}} & 8'he7)^
({{8{data_in[201][3]}} & 8'he5)^
({{8{data_in[201][4]}} & 8'he1)^
({{8{data_in[201][5]}} & 8'he9)^
({{8{data_in[201][6]}} & 8'hf9)^
({{8{data_in[201][7]}} & 8'hd9)^
({{8{data_in[202][0]}} & 8'h51)^
({{8{data_in[202][1]}} & 8'ha2)^
({{8{data_in[202][2]}} & 8'h6f)^
({{8{data_in[202][3]}} & 8'hde)^
({{8{data_in[202][4]}} & 8'h97)^
({{8{data_in[202][5]}} & 8'h5)^
({{8{data_in[202][6]}} & 8'ha)^
({{8{data_in[202][7]}} & 8'h14)^
({{8{data_in[203][0]}} & 8'h6a)^
({{8{data_in[203][1]}} & 8'hd4)^
({{8{data_in[203][2]}} & 8'h83)^
({{8{data_in[203][3]}} & 8'h2d)^
({{8{data_in[203][4]}} & 8'h5a)^
({{8{data_in[203][5]}} & 8'hb4)^
({{8{data_in[203][6]}} & 8'h43)^
({{8{data_in[203][7]}} & 8'h86)^
({{8{data_in[204][0]}} & 8'h52)^
({{8{data_in[204][1]}} & 8'ha4)^
({{8{data_in[204][2]}} & 8'h63)^
({{8{data_in[204][3]}} & 8'hc6)^
({{8{data_in[204][4]}} & 8'ha7)^
({{8{data_in[204][5]}} & 8'h65)^
({{8{data_in[204][6]}} & 8'hca)^
({{8{data_in[204][7]}} & 8'hbf)^
({{8{data_in[205][0]}} & 8'he2)^
({{8{data_in[205][1]}} & 8'hef)^
({{8{data_in[205][2]}} & 8'hf5)^
({{8{data_in[205][3]}} & 8'hc1)^
({{8{data_in[205][4]}} & 8'ha9)^
({{8{data_in[205][5]}} & 8'h79)^
({{8{data_in[205][6]}} & 8'hf2)^
({{8{data_in[205][7]}} & 8'hcf)^
({{8{data_in[206][0]}} & 8'h4)^
({{8{data_in[206][1]}} & 8'h8)^
({{8{data_in[206][2]}} & 8'h10)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[206][3]}} & 8'h20)^
({{8{data_in[206][4]}} & 8'h40)^
({{8{data_in[206][5]}} & 8'h80)^
({{8{data_in[206][6]}} & 8'h2b)^
({{8{data_in[206][7]}} & 8'h56)^
({{8{data_in[207][0]}} & 8'hea)^
({{8{data_in[207][1]}} & 8'hff)^
({{8{data_in[207][2]}} & 8'hd5)^
({{8{data_in[207][3]}} & 8'h81)^
({{8{data_in[207][4]}} & 8'h29)^
({{8{data_in[207][5]}} & 8'h52)^
({{8{data_in[207][6]}} & 8'ha4)^
({{8{data_in[207][7]}} & 8'h63)^
({{8{data_in[208][0]}} & 8'h3f)^
({{8{data_in[208][1]}} & 8'h7e)^
({{8{data_in[208][2]}} & 8'hfc)^
({{8{data_in[208][3]}} & 8'hd3)^
({{8{data_in[208][4]}} & 8'h8d)^
({{8{data_in[208][5]}} & 8'h31)^
({{8{data_in[208][6]}} & 8'h62)^
({{8{data_in[208][7]}} & 8'hc4)^
({{8{data_in[209][0]}} & 8'h75)^
({{8{data_in[209][1]}} & 8'hea)^
({{8{data_in[209][2]}} & 8'hff)^
({{8{data_in[209][3]}} & 8'hd5)^
({{8{data_in[209][4]}} & 8'h81)^
({{8{data_in[209][5]}} & 8'h29)^
({{8{data_in[209][6]}} & 8'h52)^
({{8{data_in[209][7]}} & 8'ha4)^
({{8{data_in[210][0]}} & 8'h23)^
({{8{data_in[210][1]}} & 8'h46)^
({{8{data_in[210][2]}} & 8'h8c)^
({{8{data_in[210][3]}} & 8'h33)^
({{8{data_in[210][4]}} & 8'h66)^
({{8{data_in[210][5]}} & 8'hcc)^
({{8{data_in[210][6]}} & 8'hb3)^
({{8{data_in[210][7]}} & 8'h4d)^
({{8{data_in[211][0]}} & 8'h68)^
({{8{data_in[211][1]}} & 8'hd0)^
({{8{data_in[211][2]}} & 8'h8b)^
({{8{data_in[211][3]}} & 8'h3d)^
({{8{data_in[211][4]}} & 8'h7a)^
({{8{data_in[211][5]}} & 8'hf4)^
({{8{data_in[211][6]}} & 8'hc3)^
({{8{data_in[211][7]}} & 8'had)^
({{8{data_in[212][0]}} & 8'ha5)^
({{8{data_in[212][1]}} & 8'h61)^
({{8{data_in[212][2]}} & 8'hc2)^
({{8{data_in[212][3]}} & 8'haf)^
({{8{data_in[212][4]}} & 8'h75)^
({{8{data_in[212][5]}} & 8'hea)^
({{8{data_in[212][6]}} & 8'hff)^
({{8{data_in[212][7]}} & 8'hd5)^
({{8{data_in[213][0]}} & 8'h3a)^
({{8{data_in[213][1]}} & 8'h74)^
({{8{data_in[213][2]}} & 8'he8)^
({{8{data_in[213][3]}} & 8'hfb)^
({{8{data_in[213][4]}} & 8'hdd)^
({{8{data_in[213][5]}} & 8'h91)^
({{8{data_in[213][6]}} & 8'h9)^
({{8{data_in[213][7]}} & 8'h12)^
({{8{data_in[214][0]}} & 8'h6a)^
({{8{data_in[214][1]}} & 8'hd4)^
({{8{data_in[214][2]}} & 8'h83)^
({{8{data_in[214][3]}} & 8'h2d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[214][4]}} & 8'h5a)^
({{8{data_in[214][5]}} & 8'hb4)^
({{8{data_in[214][6]}} & 8'h43)^
({{8{data_in[214][7]}} & 8'h86)^
({{8{data_in[215][0]}} & 8'h87)^
({{8{data_in[215][1]}} & 8'h25)^
({{8{data_in[215][2]}} & 8'h4a)^
({{8{data_in[215][3]}} & 8'h94)^
({{8{data_in[215][4]}} & 8'h3)^
({{8{data_in[215][5]}} & 8'h6)^
({{8{data_in[215][6]}} & 8'hc)^
({{8{data_in[215][7]}} & 8'h18)^
({{8{data_in[216][0]}} & 8'he7)^
({{8{data_in[216][1]}} & 8'he5)^
({{8{data_in[216][2]}} & 8'he1)^
({{8{data_in[216][3]}} & 8'he9)^
({{8{data_in[216][4]}} & 8'hf9)^
({{8{data_in[216][5]}} & 8'hd9)^
({{8{data_in[216][6]}} & 8'h99)^
({{8{data_in[216][7]}} & 8'h19)^
({{8{data_in[217][0]}} & 8'hc5)^
({{8{data_in[217][1]}} & 8'ha1)^
({{8{data_in[217][2]}} & 8'h69)^
({{8{data_in[217][3]}} & 8'hd2)^
({{8{data_in[217][4]}} & 8'h8f)^
({{8{data_in[217][5]}} & 8'h35)^
({{8{data_in[217][6]}} & 8'h6a)^
({{8{data_in[217][7]}} & 8'hd4)^
({{8{data_in[218][0]}} & 8'h14)^
({{8{data_in[218][1]}} & 8'h28)^
({{8{data_in[218][2]}} & 8'h50)^
({{8{data_in[218][3]}} & 8'ha0)^
({{8{data_in[218][4]}} & 8'h6b)^
({{8{data_in[218][5]}} & 8'hd6)^
({{8{data_in[218][6]}} & 8'h87)^
({{8{data_in[218][7]}} & 8'h25)^
({{8{data_in[219][0]}} & 8'h6f)^
({{8{data_in[219][1]}} & 8'hde)^
({{8{data_in[219][2]}} & 8'h97)^
({{8{data_in[219][3]}} & 8'h5)^
({{8{data_in[219][4]}} & 8'ha)^
({{8{data_in[219][5]}} & 8'h14)^
({{8{data_in[219][6]}} & 8'h28)^
({{8{data_in[219][7]}} & 8'h50)^
({{8{data_in[220][0]}} & 8'h54)^
({{8{data_in[220][1]}} & 8'ha8)^
({{8{data_in[220][2]}} & 8'h7b)^
({{8{data_in[220][3]}} & 8'hf6)^
({{8{data_in[220][4]}} & 8'hc7)^
({{8{data_in[220][5]}} & 8'ha5)^
({{8{data_in[220][6]}} & 8'h61)^
({{8{data_in[220][7]}} & 8'hc2)^
({{8{data_in[221][0]}} & 8'hc2)^
({{8{data_in[221][1]}} & 8'haf)^
({{8{data_in[221][2]}} & 8'h75)^
({{8{data_in[221][3]}} & 8'hea)^
({{8{data_in[221][4]}} & 8'hff)^
({{8{data_in[221][5]}} & 8'hd5)^
({{8{data_in[221][6]}} & 8'h81)^
({{8{data_in[221][7]}} & 8'h29)^
({{8{data_in[222][0]}} & 8'ha8)^
({{8{data_in[222][1]}} & 8'h7b)^
({{8{data_in[222][2]}} & 8'hf6)^
({{8{data_in[222][3]}} & 8'hc7)^
({{8{data_in[222][4]}} & 8'ha5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[222][5]}} & 8'h61)^
({{8{data_in[222][6]}} & 8'hc2)^
({{8{data_in[222][7]}} & 8'haf)^
({{8{data_in[223][0]}} & 8'h30)^
({{8{data_in[223][1]}} & 8'h60)^
({{8{data_in[223][2]}} & 8'hc0)^
({{8{data_in[223][3]}} & 8'hab)^
({{8{data_in[223][4]}} & 8'h7d)^
({{8{data_in[223][5]}} & 8'hfa)^
({{8{data_in[223][6]}} & 8'hdf)^
({{8{data_in[223][7]}} & 8'h95)^
({{8{data_in[224][0]}} & 8'hac)^
({{8{data_in[224][1]}} & 8'h73)^
({{8{data_in[224][2]}} & 8'he6)^
({{8{data_in[224][3]}} & 8'he7)^
({{8{data_in[224][4]}} & 8'he5)^
({{8{data_in[224][5]}} & 8'he1)^
({{8{data_in[224][6]}} & 8'he9)^
({{8{data_in[224][7]}} & 8'hf9)^
({{8{data_in[225][0]}} & 8'h9b)^
({{8{data_in[225][1]}} & 8'h1d)^
({{8{data_in[225][2]}} & 8'h3a)^
({{8{data_in[225][3]}} & 8'h74)^
({{8{data_in[225][4]}} & 8'he8)^
({{8{data_in[225][5]}} & 8'hfb)^
({{8{data_in[225][6]}} & 8'hdd)^
({{8{data_in[225][7]}} & 8'h91)^
({{8{data_in[226][0]}} & 8'hee)^
({{8{data_in[226][1]}} & 8'hf7)^
({{8{data_in[226][2]}} & 8'hc5)^
({{8{data_in[226][3]}} & 8'ha1)^
({{8{data_in[226][4]}} & 8'h69)^
({{8{data_in[226][5]}} & 8'hd2)^
({{8{data_in[226][6]}} & 8'h8f)^
({{8{data_in[226][7]}} & 8'h35)^
({{8{data_in[227][0]}} & 8'he9)^
({{8{data_in[227][1]}} & 8'hf9)^
({{8{data_in[227][2]}} & 8'hd9)^
({{8{data_in[227][3]}} & 8'h99)^
({{8{data_in[227][4]}} & 8'h19)^
({{8{data_in[227][5]}} & 8'h32)^
({{8{data_in[227][6]}} & 8'h64)^
({{8{data_in[227][7]}} & 8'hc8)^
({{8{data_in[228][0]}} & 8'ha1)^
({{8{data_in[228][1]}} & 8'h69)^
({{8{data_in[228][2]}} & 8'hd2)^
({{8{data_in[228][3]}} & 8'h8f)^
({{8{data_in[228][4]}} & 8'h35)^
({{8{data_in[228][5]}} & 8'h6a)^
({{8{data_in[228][6]}} & 8'hd4)^
({{8{data_in[228][7]}} & 8'h83)^
({{8{data_in[229][0]}} & 8'h30)^
({{8{data_in[229][1]}} & 8'h60)^
({{8{data_in[229][2]}} & 8'hc0)^
({{8{data_in[229][3]}} & 8'hab)^
({{8{data_in[229][4]}} & 8'h7d)^
({{8{data_in[229][5]}} & 8'hfa)^
({{8{data_in[229][6]}} & 8'hdf)^
({{8{data_in[229][7]}} & 8'h95)^
({{8{data_in[230][0]}} & 8'h30)^
({{8{data_in[230][1]}} & 8'h60)^
({{8{data_in[230][2]}} & 8'hc0)^
({{8{data_in[230][3]}} & 8'hab)^
({{8{data_in[230][4]}} & 8'h7d)^
({{8{data_in[230][5]}} & 8'hfa)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[230][6]}} & 8'hdf)^
({{8{data_in[230][7]}} & 8'h95)^
({{8{data_in[231][0]}} & 8'h19)^
({{8{data_in[231][1]}} & 8'h32)^
({{8{data_in[231][2]}} & 8'h64)^
({{8{data_in[231][3]}} & 8'hc8)^
({{8{data_in[231][4]}} & 8'hbb)^
({{8{data_in[231][5]}} & 8'h5d)^
({{8{data_in[231][6]}} & 8'hba)^
({{8{data_in[231][7]}} & 8'h5f)^
({{8{data_in[232][0]}} & 8'hf6)^
({{8{data_in[232][1]}} & 8'hc7)^
({{8{data_in[232][2]}} & 8'ha5)^
({{8{data_in[232][3]}} & 8'h61)^
({{8{data_in[232][4]}} & 8'hc2)^
({{8{data_in[232][5]}} & 8'haf)^
({{8{data_in[232][6]}} & 8'h75)^
({{8{data_in[232][7]}} & 8'hea)^
({{8{data_in[233][0]}} & 8'h4a)^
({{8{data_in[233][1]}} & 8'h94)^
({{8{data_in[233][2]}} & 8'h3)^
({{8{data_in[233][3]}} & 8'h6)^
({{8{data_in[233][4]}} & 8'hc)^
({{8{data_in[233][5]}} & 8'h18)^
({{8{data_in[233][6]}} & 8'h30)^
({{8{data_in[233][7]}} & 8'h60)^
({{8{data_in[234][0]}} & 8'h10)^
({{8{data_in[234][1]}} & 8'h20)^
({{8{data_in[234][2]}} & 8'h40)^
({{8{data_in[234][3]}} & 8'h80)^
({{8{data_in[234][4]}} & 8'h2b)^
({{8{data_in[234][5]}} & 8'h56)^
({{8{data_in[234][6]}} & 8'hac)^
({{8{data_in[234][7]}} & 8'h73)^
({{8{data_in[235][0]}} & 8'h9b)^
({{8{data_in[235][1]}} & 8'h1d)^
({{8{data_in[235][2]}} & 8'h3a)^
({{8{data_in[235][3]}} & 8'h74)^
({{8{data_in[235][4]}} & 8'he8)^
({{8{data_in[235][5]}} & 8'hfb)^
({{8{data_in[235][6]}} & 8'hdd)^
({{8{data_in[235][7]}} & 8'h91)^
({{8{data_in[236][0]}} & 8'h33)^
({{8{data_in[236][1]}} & 8'h66)^
({{8{data_in[236][2]}} & 8'hcc)^
({{8{data_in[236][3]}} & 8'hb3)^
({{8{data_in[236][4]}} & 8'h4d)^
({{8{data_in[236][5]}} & 8'h9a)^
({{8{data_in[236][6]}} & 8'h1f)^
({{8{data_in[236][7]}} & 8'h3e)^
({{8{data_in[237][0]}} & 8'h4a)^
({{8{data_in[237][1]}} & 8'h94)^
({{8{data_in[237][2]}} & 8'h3)^
({{8{data_in[237][3]}} & 8'h6)^
({{8{data_in[237][4]}} & 8'hc)^
({{8{data_in[237][5]}} & 8'h18)^
({{8{data_in[237][6]}} & 8'h30)^
({{8{data_in[237][7]}} & 8'h60)^
({{8{data_in[238][0]}} & 8'ha5)^
({{8{data_in[238][1]}} & 8'h61)^
({{8{data_in[238][2]}} & 8'hc2)^
({{8{data_in[238][3]}} & 8'haf)^
({{8{data_in[238][4]}} & 8'h75)^
({{8{data_in[238][5]}} & 8'hea)^
({{8{data_in[238][6]}} & 8'hff)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[238][7]}} & 8'hd5)^
({{8{data_in[239][0]}} & 8'h65)^
({{8{data_in[239][1]}} & 8'hca)^
({{8{data_in[239][2]}} & 8'hbf)^
({{8{data_in[239][3]}} & 8'h55)^
({{8{data_in[239][4]}} & 8'haa)^
({{8{data_in[239][5]}} & 8'h7f)^
({{8{data_in[239][6]}} & 8'hfe)^
({{8{data_in[239][7]}} & 8'hd7)^
({{8{data_in[240][0]}} & 8'ha)^
({{8{data_in[240][1]}} & 8'h14)^
({{8{data_in[240][2]}} & 8'h28)^
({{8{data_in[240][3]}} & 8'h50)^
({{8{data_in[240][4]}} & 8'ha0)^
({{8{data_in[240][5]}} & 8'h6b)^
({{8{data_in[240][6]}} & 8'hd6)^
({{8{data_in[240][7]}} & 8'h87)^
({{8{data_in[241][0]}} & 8'hd5)^
({{8{data_in[241][1]}} & 8'h81)^
({{8{data_in[241][2]}} & 8'h29)^
({{8{data_in[241][3]}} & 8'h52)^
({{8{data_in[241][4]}} & 8'ha4)^
({{8{data_in[241][5]}} & 8'h63)^
({{8{data_in[241][6]}} & 8'hc6)^
({{8{data_in[241][7]}} & 8'ha7);

data_out[245] = ({{8{data_in[0][0]}} & 8'h1a)^
    ({{8{data_in[0][1]}} & 8'h34)^
    ({{8{data_in[0][2]}} & 8'h68)^
    ({{8{data_in[0][3]}} & 8'hd0)^
    ({{8{data_in[0][4]}} & 8'h8b)^
    ({{8{data_in[0][5]}} & 8'h3d)^
    ({{8{data_in[0][6]}} & 8'h7a)^
    ({{8{data_in[0][7]}} & 8'hf4)^
    ({{8{data_in[1][0]}} & 8'hcb)^
    ({{8{data_in[1][1]}} & 8'hbd)^
    ({{8{data_in[1][2]}} & 8'h51)^
    ({{8{data_in[1][3]}} & 8'ha2)^
    ({{8{data_in[1][4]}} & 8'h6f)^
    ({{8{data_in[1][5]}} & 8'hde)^
    ({{8{data_in[1][6]}} & 8'h97)^
    ({{8{data_in[1][7]}} & 8'h5)^
    ({{8{data_in[2][0]}} & 8'h54)^
    ({{8{data_in[2][1]}} & 8'ha8)^
    ({{8{data_in[2][2]}} & 8'h7b)^
    ({{8{data_in[2][3]}} & 8'hf6)^
    ({{8{data_in[2][4]}} & 8'hc7)^
    ({{8{data_in[2][5]}} & 8'ha5)^
    ({{8{data_in[2][6]}} & 8'h61)^
    ({{8{data_in[2][7]}} & 8'hc2)^
    ({{8{data_in[3][0]}} & 8'hb5)^
    ({{8{data_in[3][1]}} & 8'h41)^
    ({{8{data_in[3][2]}} & 8'h82)^
    ({{8{data_in[3][3]}} & 8'h2f)^
    ({{8{data_in[3][4]}} & 8'h5e)^
    ({{8{data_in[3][5]}} & 8'hbc)^
    ({{8{data_in[3][6]}} & 8'h53)^
    ({{8{data_in[3][7]}} & 8'ha6)^
    ({{8{data_in[4][0]}} & 8'h43)^
    ({{8{data_in[4][1]}} & 8'h86)^
    ({{8{data_in[4][2]}} & 8'h27)^
    ({{8{data_in[4][3]}} & 8'h4e)^
    ({{8{data_in[4][4]}} & 8'h9c)^
    ({{8{data_in[4][5]}} & 8'h13)^
    ({{8{data_in[4][6]}} & 8'h26)^
    ({{8{data_in[4][7]}} & 8'h4c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[5][0]}} & 8'hd3)^
({{8{data_in[5][1]}} & 8'h8d)^
({{8{data_in[5][2]}} & 8'h31)^
({{8{data_in[5][3]}} & 8'h62)^
({{8{data_in[5][4]}} & 8'hc4)^
({{8{data_in[5][5]}} & 8'ha3)^
({{8{data_in[5][6]}} & 8'h6d)^
({{8{data_in[5][7]}} & 8'hda)^
({{8{data_in[6][0]}} & 8'h93)^
({{8{data_in[6][1]}} & 8'hd)^
({{8{data_in[6][2]}} & 8'h1a)^
({{8{data_in[6][3]}} & 8'h34)^
({{8{data_in[6][4]}} & 8'h68)^
({{8{data_in[6][5]}} & 8'hd0)^
({{8{data_in[6][6]}} & 8'h8b)^
({{8{data_in[6][7]}} & 8'h3d)^
({{8{data_in[7][0]}} & 8'h5c)^
({{8{data_in[7][1]}} & 8'hb8)^
({{8{data_in[7][2]}} & 8'h5b)^
({{8{data_in[7][3]}} & 8'hb6)^
({{8{data_in[7][4]}} & 8'h47)^
({{8{data_in[7][5]}} & 8'h8e)^
({{8{data_in[7][6]}} & 8'h37)^
({{8{data_in[7][7]}} & 8'h6e)^
({{8{data_in[8][0]}} & 8'h5e)^
({{8{data_in[8][1]}} & 8'hbc)^
({{8{data_in[8][2]}} & 8'h53)^
({{8{data_in[8][3]}} & 8'ha6)^
({{8{data_in[8][4]}} & 8'h67)^
({{8{data_in[8][5]}} & 8'hce)^
({{8{data_in[8][6]}} & 8'hb7)^
({{8{data_in[8][7]}} & 8'h45)^
({{8{data_in[9][0]}} & 8'h37)^
({{8{data_in[9][1]}} & 8'h6e)^
({{8{data_in[9][2]}} & 8'hdc)^
({{8{data_in[9][3]}} & 8'h93)^
({{8{data_in[9][4]}} & 8'hd)^
({{8{data_in[9][5]}} & 8'h1a)^
({{8{data_in[9][6]}} & 8'h34)^
({{8{data_in[9][7]}} & 8'h68)^
({{8{data_in[10][0]}} & 8'h68)^
({{8{data_in[10][1]}} & 8'hd0)^
({{8{data_in[10][2]}} & 8'h8b)^
({{8{data_in[10][3]}} & 8'h3d)^
({{8{data_in[10][4]}} & 8'h7a)^
({{8{data_in[10][5]}} & 8'hf4)^
({{8{data_in[10][6]}} & 8'hc3)^
({{8{data_in[10][7]}} & 8'had)^
({{8{data_in[11][0]}} & 8'h41)^
({{8{data_in[11][1]}} & 8'h82)^
({{8{data_in[11][2]}} & 8'h2f)^
({{8{data_in[11][3]}} & 8'h5e)^
({{8{data_in[11][4]}} & 8'hbc)^
({{8{data_in[11][5]}} & 8'h53)^
({{8{data_in[11][6]}} & 8'ha6)^
({{8{data_in[11][7]}} & 8'h67)^
({{8{data_in[12][0]}} & 8'hb0)^
({{8{data_in[12][1]}} & 8'h4b)^
({{8{data_in[12][2]}} & 8'h96)^
({{8{data_in[12][3]}} & 8'h7)^
({{8{data_in[12][4]}} & 8'he)^
({{8{data_in[12][5]}} & 8'h1c)^
({{8{data_in[12][6]}} & 8'h38)^
({{8{data_in[12][7]}} & 8'h70)^
({{8{data_in[13][0]}} & 8'h4e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[13][1]}} & 8'h9c)^
({{8{data_in[13][2]}} & 8'h13)^
({{8{data_in[13][3]}} & 8'h26)^
({{8{data_in[13][4]}} & 8'h4c)^
({{8{data_in[13][5]}} & 8'h98)^
({{8{data_in[13][6]}} & 8'h1b)^
({{8{data_in[13][7]}} & 8'h36)^
({{8{data_in[14][0]}} & 8'h44)^
({{8{data_in[14][1]}} & 8'h88)^
({{8{data_in[14][2]}} & 8'h3b)^
({{8{data_in[14][3]}} & 8'h76)^
({{8{data_in[14][4]}} & 8'hec)^
({{8{data_in[14][5]}} & 8'hf3)^
({{8{data_in[14][6]}} & 8'hcd)^
({{8{data_in[14][7]}} & 8'hb1)^
({{8{data_in[15][0]}} & 8'h9d)^
({{8{data_in[15][1]}} & 8'h11)^
({{8{data_in[15][2]}} & 8'h22)^
({{8{data_in[15][3]}} & 8'h44)^
({{8{data_in[15][4]}} & 8'h88)^
({{8{data_in[15][5]}} & 8'h3b)^
({{8{data_in[15][6]}} & 8'h76)^
({{8{data_in[15][7]}} & 8'hec)^
({{8{data_in[16][0]}} & 8'h42)^
({{8{data_in[16][1]}} & 8'h84)^
({{8{data_in[16][2]}} & 8'h23)^
({{8{data_in[16][3]}} & 8'h46)^
({{8{data_in[16][4]}} & 8'h8c)^
({{8{data_in[16][5]}} & 8'h33)^
({{8{data_in[16][6]}} & 8'h66)^
({{8{data_in[16][7]}} & 8'hcc)^
({{8{data_in[17][0]}} & 8'hc0)^
({{8{data_in[17][1]}} & 8'hab)^
({{8{data_in[17][2]}} & 8'h7d)^
({{8{data_in[17][3]}} & 8'hfa)^
({{8{data_in[17][4]}} & 8'hdf)^
({{8{data_in[17][5]}} & 8'h95)^
({{8{data_in[17][6]}} & 8'h1)^
({{8{data_in[17][7]}} & 8'h2)^
({{8{data_in[18][0]}} & 8'hc)^
({{8{data_in[18][1]}} & 8'h18)^
({{8{data_in[18][2]}} & 8'h30)^
({{8{data_in[18][3]}} & 8'h60)^
({{8{data_in[18][4]}} & 8'hc0)^
({{8{data_in[18][5]}} & 8'hab)^
({{8{data_in[18][6]}} & 8'h7d)^
({{8{data_in[18][7]}} & 8'hfa)^
({{8{data_in[19][0]}} & 8'hd2)^
({{8{data_in[19][1]}} & 8'h8f)^
({{8{data_in[19][2]}} & 8'h35)^
({{8{data_in[19][3]}} & 8'h6a)^
({{8{data_in[19][4]}} & 8'hd4)^
({{8{data_in[19][5]}} & 8'h83)^
({{8{data_in[19][6]}} & 8'h2d)^
({{8{data_in[19][7]}} & 8'h5a)^
({{8{data_in[20][0]}} & 8'h80)^
({{8{data_in[20][1]}} & 8'h2b)^
({{8{data_in[20][2]}} & 8'h56)^
({{8{data_in[20][3]}} & 8'hac)^
({{8{data_in[20][4]}} & 8'h73)^
({{8{data_in[20][5]}} & 8'he6)^
({{8{data_in[20][6]}} & 8'he7)^
({{8{data_in[20][7]}} & 8'he5)^
({{8{data_in[21][0]}} & 8'hcb)^
({{8{data_in[21][1]}} & 8'hbd)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[21][2]}} & 8'h51)^
({{8{data_in[21][3]}} & 8'ha2)^
({{8{data_in[21][4]}} & 8'h6f)^
({{8{data_in[21][5]}} & 8'hde)^
({{8{data_in[21][6]}} & 8'h97)^
({{8{data_in[21][7]}} & 8'h5)^
({{8{data_in[22][0]}} & 8'he9)^
({{8{data_in[22][1]}} & 8'hf9)^
({{8{data_in[22][2]}} & 8'hd9)^
({{8{data_in[22][3]}} & 8'h99)^
({{8{data_in[22][4]}} & 8'h19)^
({{8{data_in[22][5]}} & 8'h32)^
({{8{data_in[22][6]}} & 8'h64)^
({{8{data_in[22][7]}} & 8'hc8)^
({{8{data_in[23][0]}} & 8'hbe)^
({{8{data_in[23][1]}} & 8'h57)^
({{8{data_in[23][2]}} & 8'hae)^
({{8{data_in[23][3]}} & 8'h77)^
({{8{data_in[23][4]}} & 8'hee)^
({{8{data_in[23][5]}} & 8'hf7)^
({{8{data_in[23][6]}} & 8'hc5)^
({{8{data_in[23][7]}} & 8'ha1)^
({{8{data_in[24][0]}} & 8'had)^
({{8{data_in[24][1]}} & 8'h71)^
({{8{data_in[24][2]}} & 8'he2)^
({{8{data_in[24][3]}} & 8'hef)^
({{8{data_in[24][4]}} & 8'hf5)^
({{8{data_in[24][5]}} & 8'hc1)^
({{8{data_in[24][6]}} & 8'ha9)^
({{8{data_in[24][7]}} & 8'h79)^
({{8{data_in[25][0]}} & 8'hdd)^
({{8{data_in[25][1]}} & 8'h91)^
({{8{data_in[25][2]}} & 8'h9)^
({{8{data_in[25][3]}} & 8'h12)^
({{8{data_in[25][4]}} & 8'h24)^
({{8{data_in[25][5]}} & 8'h48)^
({{8{data_in[25][6]}} & 8'h90)^
({{8{data_in[25][7]}} & 8'hb)^
({{8{data_in[26][0]}} & 8'hcf)^
({{8{data_in[26][1]}} & 8'hb5)^
({{8{data_in[26][2]}} & 8'h41)^
({{8{data_in[26][3]}} & 8'h82)^
({{8{data_in[26][4]}} & 8'h2f)^
({{8{data_in[26][5]}} & 8'h5e)^
({{8{data_in[26][6]}} & 8'hbc)^
({{8{data_in[26][7]}} & 8'h53)^
({{8{data_in[27][0]}} & 8'h9e)^
({{8{data_in[27][1]}} & 8'h17)^
({{8{data_in[27][2]}} & 8'h2e)^
({{8{data_in[27][3]}} & 8'h5c)^
({{8{data_in[27][4]}} & 8'hb8)^
({{8{data_in[27][5]}} & 8'h5b)^
({{8{data_in[27][6]}} & 8'hb6)^
({{8{data_in[27][7]}} & 8'h47)^
({{8{data_in[28][0]}} & 8'h12)^
({{8{data_in[28][1]}} & 8'h24)^
({{8{data_in[28][2]}} & 8'h48)^
({{8{data_in[28][3]}} & 8'h90)^
({{8{data_in[28][4]}} & 8'hb)^
({{8{data_in[28][5]}} & 8'h16)^
({{8{data_in[28][6]}} & 8'h2c)^
({{8{data_in[28][7]}} & 8'h58)^
({{8{data_in[29][0]}} & 8'h70)^
({{8{data_in[29][1]}} & 8'he0)^
({{8{data_in[29][2]}} & 8'heb)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[29][3]}} & 8'hfd)^
({{8{data_in[29][4]}} & 8'hd1)^
({{8{data_in[29][5]}} & 8'h89)^
({{8{data_in[29][6]}} & 8'h39)^
({{8{data_in[29][7]}} & 8'h72)^
({{8{data_in[30][0]}} & 8'ha8)^
({{8{data_in[30][1]}} & 8'h7b)^
({{8{data_in[30][2]}} & 8'hf6)^
({{8{data_in[30][3]}} & 8'hc7)^
({{8{data_in[30][4]}} & 8'ha5)^
({{8{data_in[30][5]}} & 8'h61)^
({{8{data_in[30][6]}} & 8'hc2)^
({{8{data_in[30][7]}} & 8'haf)^
({{8{data_in[31][0]}} & 8'h19)^
({{8{data_in[31][1]}} & 8'h32)^
({{8{data_in[31][2]}} & 8'h64)^
({{8{data_in[31][3]}} & 8'hc8)^
({{8{data_in[31][4]}} & 8'hbb)^
({{8{data_in[31][5]}} & 8'h5d)^
({{8{data_in[31][6]}} & 8'hba)^
({{8{data_in[31][7]}} & 8'h5f)^
({{8{data_in[32][0]}} & 8'h3a)^
({{8{data_in[32][1]}} & 8'h74)^
({{8{data_in[32][2]}} & 8'he8)^
({{8{data_in[32][3]}} & 8'hfb)^
({{8{data_in[32][4]}} & 8'hdd)^
({{8{data_in[32][5]}} & 8'h91)^
({{8{data_in[32][6]}} & 8'h9)^
({{8{data_in[32][7]}} & 8'h12)^
({{8{data_in[33][0]}} & 8'hd5)^
({{8{data_in[33][1]}} & 8'h81)^
({{8{data_in[33][2]}} & 8'h29)^
({{8{data_in[33][3]}} & 8'h52)^
({{8{data_in[33][4]}} & 8'ha4)^
({{8{data_in[33][5]}} & 8'h63)^
({{8{data_in[33][6]}} & 8'hc6)^
({{8{data_in[33][7]}} & 8'ha7)^
({{8{data_in[34][0]}} & 8'hf1)^
({{8{data_in[34][1]}} & 8'hc9)^
({{8{data_in[34][2]}} & 8'hb9)^
({{8{data_in[34][3]}} & 8'h59)^
({{8{data_in[34][4]}} & 8'hb2)^
({{8{data_in[34][5]}} & 8'h4f)^
({{8{data_in[34][6]}} & 8'h9e)^
({{8{data_in[34][7]}} & 8'h17)^
({{8{data_in[35][0]}} & 8'he2)^
({{8{data_in[35][1]}} & 8'hef)^
({{8{data_in[35][2]}} & 8'hf5)^
({{8{data_in[35][3]}} & 8'hc1)^
({{8{data_in[35][4]}} & 8'ha9)^
({{8{data_in[35][5]}} & 8'h79)^
({{8{data_in[35][6]}} & 8'hf2)^
({{8{data_in[35][7]}} & 8'hcf)^
({{8{data_in[36][0]}} & 8'he9)^
({{8{data_in[36][1]}} & 8'hf9)^
({{8{data_in[36][2]}} & 8'hd9)^
({{8{data_in[36][3]}} & 8'h99)^
({{8{data_in[36][4]}} & 8'h19)^
({{8{data_in[36][5]}} & 8'h32)^
({{8{data_in[36][6]}} & 8'h64)^
({{8{data_in[36][7]}} & 8'hc8)^
({{8{data_in[37][0]}} & 8'h64)^
({{8{data_in[37][1]}} & 8'hc8)^
({{8{data_in[37][2]}} & 8'hbb)^
({{8{data_in[37][3]}} & 8'h5d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[37][4]}} & 8'hba)^
({{8{data_in[37][5]}} & 8'h5f)^
({{8{data_in[37][6]}} & 8'hbe)^
({{8{data_in[37][7]}} & 8'h57)^
({{8{data_in[38][0]}} & 8'hf3)^
({{8{data_in[38][1]}} & 8'hcd)^
({{8{data_in[38][2]}} & 8'hb1)^
({{8{data_in[38][3]}} & 8'h49)^
({{8{data_in[38][4]}} & 8'h92)^
({{8{data_in[38][5]}} & 8'hf)^
({{8{data_in[38][6]}} & 8'h1e)^
({{8{data_in[38][7]}} & 8'h3c)^
({{8{data_in[39][0]}} & 8'h1c)^
({{8{data_in[39][1]}} & 8'h38)^
({{8{data_in[39][2]}} & 8'h70)^
({{8{data_in[39][3]}} & 8'he0)^
({{8{data_in[39][4]}} & 8'heb)^
({{8{data_in[39][5]}} & 8'hfd)^
({{8{data_in[39][6]}} & 8'hd1)^
({{8{data_in[39][7]}} & 8'h89)^
({{8{data_in[40][0]}} & 8'he3)^
({{8{data_in[40][1]}} & 8'hed)^
({{8{data_in[40][2]}} & 8'hf1)^
({{8{data_in[40][3]}} & 8'hc9)^
({{8{data_in[40][4]}} & 8'hb9)^
({{8{data_in[40][5]}} & 8'h59)^
({{8{data_in[40][6]}} & 8'hb2)^
({{8{data_in[40][7]}} & 8'h4f)^
({{8{data_in[41][0]}} & 8'h3a)^
({{8{data_in[41][1]}} & 8'h74)^
({{8{data_in[41][2]}} & 8'he8)^
({{8{data_in[41][3]}} & 8'hfb)^
({{8{data_in[41][4]}} & 8'hdd)^
({{8{data_in[41][5]}} & 8'h91)^
({{8{data_in[41][6]}} & 8'h9)^
({{8{data_in[41][7]}} & 8'h12)^
({{8{data_in[42][0]}} & 8'h91)^
({{8{data_in[42][1]}} & 8'h9)^
({{8{data_in[42][2]}} & 8'h12)^
({{8{data_in[42][3]}} & 8'h24)^
({{8{data_in[42][4]}} & 8'h48)^
({{8{data_in[42][5]}} & 8'h90)^
({{8{data_in[42][6]}} & 8'hb)^
({{8{data_in[42][7]}} & 8'h16)^
({{8{data_in[43][0]}} & 8'h82)^
({{8{data_in[43][1]}} & 8'h2f)^
({{8{data_in[43][2]}} & 8'h5e)^
({{8{data_in[43][3]}} & 8'hbc)^
({{8{data_in[43][4]}} & 8'h53)^
({{8{data_in[43][5]}} & 8'ha6)^
({{8{data_in[43][6]}} & 8'h67)^
({{8{data_in[43][7]}} & 8'hce)^
({{8{data_in[44][0]}} & 8'hee)^
({{8{data_in[44][1]}} & 8'hf7)^
({{8{data_in[44][2]}} & 8'hc5)^
({{8{data_in[44][3]}} & 8'ha1)^
({{8{data_in[44][4]}} & 8'h69)^
({{8{data_in[44][5]}} & 8'hd2)^
({{8{data_in[44][6]}} & 8'h8f)^
({{8{data_in[44][7]}} & 8'h35)^
({{8{data_in[45][0]}} & 8'he9)^
({{8{data_in[45][1]}} & 8'hf9)^
({{8{data_in[45][2]}} & 8'hd9)^
({{8{data_in[45][3]}} & 8'h99)^
({{8{data_in[45][4]}} & 8'h19})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[45][5]}} & 8'h32)^
({{8{data_in[45][6]}} & 8'h64)^
({{8{data_in[45][7]}} & 8'hc8)^
({{8{data_in[46][0]}} & 8'hb4)^
({{8{data_in[46][1]}} & 8'h43)^
({{8{data_in[46][2]}} & 8'h86)^
({{8{data_in[46][3]}} & 8'h27)^
({{8{data_in[46][4]}} & 8'h4e)^
({{8{data_in[46][5]}} & 8'h9c)^
({{8{data_in[46][6]}} & 8'h13)^
({{8{data_in[46][7]}} & 8'h26)^
({{8{data_in[47][0]}} & 8'h58)^
({{8{data_in[47][1]}} & 8'hb0)^
({{8{data_in[47][2]}} & 8'h4b)^
({{8{data_in[47][3]}} & 8'h96)^
({{8{data_in[47][4]}} & 8'h7)^
({{8{data_in[47][5]}} & 8'he)^
({{8{data_in[47][6]}} & 8'h1c)^
({{8{data_in[47][7]}} & 8'h38)^
({{8{data_in[48][0]}} & 8'hf7)^
({{8{data_in[48][1]}} & 8'hc5)^
({{8{data_in[48][2]}} & 8'ha1)^
({{8{data_in[48][3]}} & 8'h69)^
({{8{data_in[48][4]}} & 8'hd2)^
({{8{data_in[48][5]}} & 8'h8f)^
({{8{data_in[48][6]}} & 8'h35)^
({{8{data_in[48][7]}} & 8'h6a)^
({{8{data_in[49][0]}} & 8'h6c)^
({{8{data_in[49][1]}} & 8'hd8)^
({{8{data_in[49][2]}} & 8'h9b)^
({{8{data_in[49][3]}} & 8'h1d)^
({{8{data_in[49][4]}} & 8'h3a)^
({{8{data_in[49][5]}} & 8'h74)^
({{8{data_in[49][6]}} & 8'he8)^
({{8{data_in[49][7]}} & 8'hfb)^
({{8{data_in[50][0]}} & 8'hbb)^
({{8{data_in[50][1]}} & 8'h5d)^
({{8{data_in[50][2]}} & 8'hba)^
({{8{data_in[50][3]}} & 8'h5f)^
({{8{data_in[50][4]}} & 8'hbe)^
({{8{data_in[50][5]}} & 8'h57)^
({{8{data_in[50][6]}} & 8'hae)^
({{8{data_in[50][7]}} & 8'h77)^
({{8{data_in[51][0]}} & 8'hc8)^
({{8{data_in[51][1]}} & 8'hbb)^
({{8{data_in[51][2]}} & 8'h5d)^
({{8{data_in[51][3]}} & 8'hba)^
({{8{data_in[51][4]}} & 8'h5f)^
({{8{data_in[51][5]}} & 8'hbe)^
({{8{data_in[51][6]}} & 8'h57)^
({{8{data_in[51][7]}} & 8'hae)^
({{8{data_in[52][0]}} & 8'hb9)^
({{8{data_in[52][1]}} & 8'h59)^
({{8{data_in[52][2]}} & 8'hb2)^
({{8{data_in[52][3]}} & 8'h4f)^
({{8{data_in[52][4]}} & 8'h9e)^
({{8{data_in[52][5]}} & 8'h17)^
({{8{data_in[52][6]}} & 8'h2e)^
({{8{data_in[52][7]}} & 8'h5c)^
({{8{data_in[53][0]}} & 8'h55)^
({{8{data_in[53][1]}} & 8'haa)^
({{8{data_in[53][2]}} & 8'h7f)^
({{8{data_in[53][3]}} & 8'hfe)^
({{8{data_in[53][4]}} & 8'hd7)^
({{8{data_in[53][5]}} & 8'h85)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[53][6]}} & 8'h21)^
({{8{data_in[53][7]}} & 8'h42)^
({{8{data_in[54][0]}} & 8'hf2)^
({{8{data_in[54][1]}} & 8'hcf)^
({{8{data_in[54][2]}} & 8'hb5)^
({{8{data_in[54][3]}} & 8'h41)^
({{8{data_in[54][4]}} & 8'h82)^
({{8{data_in[54][5]}} & 8'h2f)^
({{8{data_in[54][6]}} & 8'h5e)^
({{8{data_in[54][7]}} & 8'hbc)^
({{8{data_in[55][0]}} & 8'h93)^
({{8{data_in[55][1]}} & 8'hd)^
({{8{data_in[55][2]}} & 8'h1a)^
({{8{data_in[55][3]}} & 8'h34)^
({{8{data_in[55][4]}} & 8'h68)^
({{8{data_in[55][5]}} & 8'hd0)^
({{8{data_in[55][6]}} & 8'h8b)^
({{8{data_in[55][7]}} & 8'h3d)^
({{8{data_in[56][0]}} & 8'h99)^
({{8{data_in[56][1]}} & 8'h19)^
({{8{data_in[56][2]}} & 8'h32)^
({{8{data_in[56][3]}} & 8'h64)^
({{8{data_in[56][4]}} & 8'hc8)^
({{8{data_in[56][5]}} & 8'hbb)^
({{8{data_in[56][6]}} & 8'h5d)^
({{8{data_in[56][7]}} & 8'hba)^
({{8{data_in[57][0]}} & 8'h4f)^
({{8{data_in[57][1]}} & 8'h9e)^
({{8{data_in[57][2]}} & 8'h17)^
({{8{data_in[57][3]}} & 8'h2e)^
({{8{data_in[57][4]}} & 8'h5c)^
({{8{data_in[57][5]}} & 8'hb8)^
({{8{data_in[57][6]}} & 8'h5b)^
({{8{data_in[57][7]}} & 8'hb6)^
({{8{data_in[58][0]}} & 8'h44)^
({{8{data_in[58][1]}} & 8'h88)^
({{8{data_in[58][2]}} & 8'h3b)^
({{8{data_in[58][3]}} & 8'h76)^
({{8{data_in[58][4]}} & 8'hec)^
({{8{data_in[58][5]}} & 8'hf3)^
({{8{data_in[58][6]}} & 8'hcd)^
({{8{data_in[58][7]}} & 8'hb1)^
({{8{data_in[59][0]}} & 8'hb5)^
({{8{data_in[59][1]}} & 8'h41)^
({{8{data_in[59][2]}} & 8'h82)^
({{8{data_in[59][3]}} & 8'h2f)^
({{8{data_in[59][4]}} & 8'h5e)^
({{8{data_in[59][5]}} & 8'hbc)^
({{8{data_in[59][6]}} & 8'h53)^
({{8{data_in[59][7]}} & 8'ha6)^
({{8{data_in[60][0]}} & 8'hca)^
({{8{data_in[60][1]}} & 8'hbf)^
({{8{data_in[60][2]}} & 8'h55)^
({{8{data_in[60][3]}} & 8'haa)^
({{8{data_in[60][4]}} & 8'h7f)^
({{8{data_in[60][5]}} & 8'hfe)^
({{8{data_in[60][6]}} & 8'hd7)^
({{8{data_in[60][7]}} & 8'h85)^
({{8{data_in[61][0]}} & 8'haf)^
({{8{data_in[61][1]}} & 8'h75)^
({{8{data_in[61][2]}} & 8'hea)^
({{8{data_in[61][3]}} & 8'hff)^
({{8{data_in[61][4]}} & 8'hd5)^
({{8{data_in[61][5]}} & 8'h81)^
({{8{data_in[61][6]}} & 8'h29})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[61][7]}} & 8'h52)^
({{8{data_in[62][0]}} & 8'h8d)^
({{8{data_in[62][1]}} & 8'h31)^
({{8{data_in[62][2]}} & 8'h62)^
({{8{data_in[62][3]}} & 8'hc4)^
({{8{data_in[62][4]}} & 8'ha3)^
({{8{data_in[62][5]}} & 8'h6d)^
({{8{data_in[62][6]}} & 8'hda)^
({{8{data_in[62][7]}} & 8'h9f)^
({{8{data_in[63][0]}} & 8'h67)^
({{8{data_in[63][1]}} & 8'hce)^
({{8{data_in[63][2]}} & 8'hb7)^
({{8{data_in[63][3]}} & 8'h45)^
({{8{data_in[63][4]}} & 8'h8a)^
({{8{data_in[63][5]}} & 8'h3f)^
({{8{data_in[63][6]}} & 8'h7e)^
({{8{data_in[63][7]}} & 8'hfc)^
({{8{data_in[64][0]}} & 8'hac)^
({{8{data_in[64][1]}} & 8'h73)^
({{8{data_in[64][2]}} & 8'he6)^
({{8{data_in[64][3]}} & 8'he7)^
({{8{data_in[64][4]}} & 8'he5)^
({{8{data_in[64][5]}} & 8'he1)^
({{8{data_in[64][6]}} & 8'he9)^
({{8{data_in[64][7]}} & 8'hf9)^
({{8{data_in[65][0]}} & 8'h16)^
({{8{data_in[65][1]}} & 8'h2c)^
({{8{data_in[65][2]}} & 8'h58)^
({{8{data_in[65][3]}} & 8'hb0)^
({{8{data_in[65][4]}} & 8'h4b)^
({{8{data_in[65][5]}} & 8'h96)^
({{8{data_in[65][6]}} & 8'h7)^
({{8{data_in[65][7]}} & 8'he)^
({{8{data_in[66][0]}} & 8'h21)^
({{8{data_in[66][1]}} & 8'h42)^
({{8{data_in[66][2]}} & 8'h84)^
({{8{data_in[66][3]}} & 8'h23)^
({{8{data_in[66][4]}} & 8'h46)^
({{8{data_in[66][5]}} & 8'h8c)^
({{8{data_in[66][6]}} & 8'h33)^
({{8{data_in[66][7]}} & 8'h66)^
({{8{data_in[67][0]}} & 8'h4e)^
({{8{data_in[67][1]}} & 8'h9c)^
({{8{data_in[67][2]}} & 8'h13)^
({{8{data_in[67][3]}} & 8'h26)^
({{8{data_in[67][4]}} & 8'h4c)^
({{8{data_in[67][5]}} & 8'h98)^
({{8{data_in[67][6]}} & 8'h1b)^
({{8{data_in[67][7]}} & 8'h36)^
({{8{data_in[68][0]}} & 8'h97)^
({{8{data_in[68][1]}} & 8'h5)^
({{8{data_in[68][2]}} & 8'ha)^
({{8{data_in[68][3]}} & 8'h14)^
({{8{data_in[68][4]}} & 8'h28)^
({{8{data_in[68][5]}} & 8'h50)^
({{8{data_in[68][6]}} & 8'ha0)^
({{8{data_in[68][7]}} & 8'h6b)^
({{8{data_in[69][0]}} & 8'hbb)^
({{8{data_in[69][1]}} & 8'h5d)^
({{8{data_in[69][2]}} & 8'hba)^
({{8{data_in[69][3]}} & 8'h5f)^
({{8{data_in[69][4]}} & 8'hbe)^
({{8{data_in[69][5]}} & 8'h57)^
({{8{data_in[69][6]}} & 8'hae)^
({{8{data_in[69][7]}} & 8'h77)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[70][0]}} & 8'h57)^
({{8{data_in[70][1]}} & 8'hae)^
({{8{data_in[70][2]}} & 8'h77)^
({{8{data_in[70][3]}} & 8'hee)^
({{8{data_in[70][4]}} & 8'hf7)^
({{8{data_in[70][5]}} & 8'hc5)^
({{8{data_in[70][6]}} & 8'ha1)^
({{8{data_in[70][7]}} & 8'h69)^
({{8{data_in[71][0]}} & 8'h9e)^
({{8{data_in[71][1]}} & 8'h17)^
({{8{data_in[71][2]}} & 8'h2e)^
({{8{data_in[71][3]}} & 8'h5c)^
({{8{data_in[71][4]}} & 8'hb8)^
({{8{data_in[71][5]}} & 8'h5b)^
({{8{data_in[71][6]}} & 8'hb6)^
({{8{data_in[71][7]}} & 8'h47)^
({{8{data_in[72][0]}} & 8'h65)^
({{8{data_in[72][1]}} & 8'hca)^
({{8{data_in[72][2]}} & 8'hbf)^
({{8{data_in[72][3]}} & 8'h55)^
({{8{data_in[72][4]}} & 8'haa)^
({{8{data_in[72][5]}} & 8'h7f)^
({{8{data_in[72][6]}} & 8'hfe)^
({{8{data_in[72][7]}} & 8'hd7)^
({{8{data_in[73][0]}} & 8'h3a)^
({{8{data_in[73][1]}} & 8'h74)^
({{8{data_in[73][2]}} & 8'he8)^
({{8{data_in[73][3]}} & 8'hfb)^
({{8{data_in[73][4]}} & 8'hdd)^
({{8{data_in[73][5]}} & 8'h91)^
({{8{data_in[73][6]}} & 8'h9)^
({{8{data_in[73][7]}} & 8'h12)^
({{8{data_in[74][0]}} & 8'h3f)^
({{8{data_in[74][1]}} & 8'h7e)^
({{8{data_in[74][2]}} & 8'hfc)^
({{8{data_in[74][3]}} & 8'hd3)^
({{8{data_in[74][4]}} & 8'h8d)^
({{8{data_in[74][5]}} & 8'h31)^
({{8{data_in[74][6]}} & 8'h62)^
({{8{data_in[74][7]}} & 8'hc4)^
({{8{data_in[75][0]}} & 8'hd6)^
({{8{data_in[75][1]}} & 8'h87)^
({{8{data_in[75][2]}} & 8'h25)^
({{8{data_in[75][3]}} & 8'h4a)^
({{8{data_in[75][4]}} & 8'h94)^
({{8{data_in[75][5]}} & 8'h3)^
({{8{data_in[75][6]}} & 8'h6)^
({{8{data_in[75][7]}} & 8'hc)^
({{8{data_in[76][0]}} & 8'h59)^
({{8{data_in[76][1]}} & 8'hb2)^
({{8{data_in[76][2]}} & 8'h4f)^
({{8{data_in[76][3]}} & 8'h9e)^
({{8{data_in[76][4]}} & 8'h17)^
({{8{data_in[76][5]}} & 8'h2e)^
({{8{data_in[76][6]}} & 8'h5c)^
({{8{data_in[76][7]}} & 8'hb8)^
({{8{data_in[77][0]}} & 8'hf7)^
({{8{data_in[77][1]}} & 8'hc5)^
({{8{data_in[77][2]}} & 8'ha1)^
({{8{data_in[77][3]}} & 8'h69)^
({{8{data_in[77][4]}} & 8'hd2)^
({{8{data_in[77][5]}} & 8'h8f)^
({{8{data_in[77][6]}} & 8'h35)^
({{8{data_in[77][7]}} & 8'h6a)^
({{8{data_in[78][0]}} & 8'h1f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[78][1]}} & 8'h3e)^
({{8{data_in[78][2]}} & 8'h7c)^
({{8{data_in[78][3]}} & 8'hf8)^
({{8{data_in[78][4]}} & 8'hdb)^
({{8{data_in[78][5]}} & 8'h9d)^
({{8{data_in[78][6]}} & 8'h11)^
({{8{data_in[78][7]}} & 8'h22)^
({{8{data_in[79][0]}} & 8'h88)^
({{8{data_in[79][1]}} & 8'h3b)^
({{8{data_in[79][2]}} & 8'h76)^
({{8{data_in[79][3]}} & 8'hec)^
({{8{data_in[79][4]}} & 8'hf3)^
({{8{data_in[79][5]}} & 8'hcd)^
({{8{data_in[79][6]}} & 8'hb1)^
({{8{data_in[79][7]}} & 8'h49)^
({{8{data_in[80][0]}} & 8'h8b)^
({{8{data_in[80][1]}} & 8'h3d)^
({{8{data_in[80][2]}} & 8'h7a)^
({{8{data_in[80][3]}} & 8'hf4)^
({{8{data_in[80][4]}} & 8'hc3)^
({{8{data_in[80][5]}} & 8'had)^
({{8{data_in[80][6]}} & 8'h71)^
({{8{data_in[80][7]}} & 8'he2)^
({{8{data_in[81][0]}} & 8'h2)^
({{8{data_in[81][1]}} & 8'h4)^
({{8{data_in[81][2]}} & 8'h8)^
({{8{data_in[81][3]}} & 8'h10)^
({{8{data_in[81][4]}} & 8'h20)^
({{8{data_in[81][5]}} & 8'h40)^
({{8{data_in[81][6]}} & 8'h80)^
({{8{data_in[81][7]}} & 8'h2b)^
({{8{data_in[82][0]}} & 8'h31)^
({{8{data_in[82][1]}} & 8'h62)^
({{8{data_in[82][2]}} & 8'hc4)^
({{8{data_in[82][3]}} & 8'ha3)^
({{8{data_in[82][4]}} & 8'h6d)^
({{8{data_in[82][5]}} & 8'hda)^
({{8{data_in[82][6]}} & 8'h9f)^
({{8{data_in[82][7]}} & 8'h15)^
({{8{data_in[83][0]}} & 8'hca)^
({{8{data_in[83][1]}} & 8'hbf)^
({{8{data_in[83][2]}} & 8'h55)^
({{8{data_in[83][3]}} & 8'haa)^
({{8{data_in[83][4]}} & 8'h7f)^
({{8{data_in[83][5]}} & 8'hfe)^
({{8{data_in[83][6]}} & 8'hd7)^
({{8{data_in[83][7]}} & 8'h85)^
({{8{data_in[84][0]}} & 8'h96)^
({{8{data_in[84][1]}} & 8'h7)^
({{8{data_in[84][2]}} & 8'he)^
({{8{data_in[84][3]}} & 8'h1c)^
({{8{data_in[84][4]}} & 8'h38)^
({{8{data_in[84][5]}} & 8'h70)^
({{8{data_in[84][6]}} & 8'he0)^
({{8{data_in[84][7]}} & 8'heb)^
({{8{data_in[85][0]}} & 8'h30)^
({{8{data_in[85][1]}} & 8'h60)^
({{8{data_in[85][2]}} & 8'hc0)^
({{8{data_in[85][3]}} & 8'hab)^
({{8{data_in[85][4]}} & 8'h7d)^
({{8{data_in[85][5]}} & 8'hfa)^
({{8{data_in[85][6]}} & 8'hdf)^
({{8{data_in[85][7]}} & 8'h95)^
({{8{data_in[86][0]}} & 8'hf)^
({{8{data_in[86][1]}} & 8'h1e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[86][2]}} & 8'h3c)^
({{8{data_in[86][3]}} & 8'h78)^
({{8{data_in[86][4]}} & 8'hf0)^
({{8{data_in[86][5]}} & 8'hcb)^
({{8{data_in[86][6]}} & 8'hbd)^
({{8{data_in[86][7]}} & 8'h51)^
({{8{data_in[87][0]}} & 8'h7b)^
({{8{data_in[87][1]}} & 8'hf6)^
({{8{data_in[87][2]}} & 8'hc7)^
({{8{data_in[87][3]}} & 8'ha5)^
({{8{data_in[87][4]}} & 8'h61)^
({{8{data_in[87][5]}} & 8'hc2)^
({{8{data_in[87][6]}} & 8'haf)^
({{8{data_in[87][7]}} & 8'h75)^
({{8{data_in[88][0]}} & 8'h2d)^
({{8{data_in[88][1]}} & 8'h5a)^
({{8{data_in[88][2]}} & 8'hb4)^
({{8{data_in[88][3]}} & 8'h43)^
({{8{data_in[88][4]}} & 8'h86)^
({{8{data_in[88][5]}} & 8'h27)^
({{8{data_in[88][6]}} & 8'h4e)^
({{8{data_in[88][7]}} & 8'h9c)^
({{8{data_in[89][0]}} & 8'he)^
({{8{data_in[89][1]}} & 8'h1c)^
({{8{data_in[89][2]}} & 8'h38)^
({{8{data_in[89][3]}} & 8'h70)^
({{8{data_in[89][4]}} & 8'he0)^
({{8{data_in[89][5]}} & 8'heb)^
({{8{data_in[89][6]}} & 8'hfd)^
({{8{data_in[89][7]}} & 8'hd1)^
({{8{data_in[90][0]}} & 8'hf0)^
({{8{data_in[90][1]}} & 8'hcb)^
({{8{data_in[90][2]}} & 8'hbd)^
({{8{data_in[90][3]}} & 8'h51)^
({{8{data_in[90][4]}} & 8'ha2)^
({{8{data_in[90][5]}} & 8'h6f)^
({{8{data_in[90][6]}} & 8'hde)^
({{8{data_in[90][7]}} & 8'h97)^
({{8{data_in[91][0]}} & 8'h4f)^
({{8{data_in[91][1]}} & 8'h9e)^
({{8{data_in[91][2]}} & 8'h17)^
({{8{data_in[91][3]}} & 8'h2e)^
({{8{data_in[91][4]}} & 8'h5c)^
({{8{data_in[91][5]}} & 8'hb8)^
({{8{data_in[91][6]}} & 8'h5b)^
({{8{data_in[91][7]}} & 8'hb6)^
({{8{data_in[92][0]}} & 8'h45)^
({{8{data_in[92][1]}} & 8'h8a)^
({{8{data_in[92][2]}} & 8'h3f)^
({{8{data_in[92][3]}} & 8'h7e)^
({{8{data_in[92][4]}} & 8'hfc)^
({{8{data_in[92][5]}} & 8'hd3)^
({{8{data_in[92][6]}} & 8'h8d)^
({{8{data_in[92][7]}} & 8'h31)^
({{8{data_in[93][0]}} & 8'hd9)^
({{8{data_in[93][1]}} & 8'h99)^
({{8{data_in[93][2]}} & 8'h19)^
({{8{data_in[93][3]}} & 8'h32)^
({{8{data_in[93][4]}} & 8'h64)^
({{8{data_in[93][5]}} & 8'hc8)^
({{8{data_in[93][6]}} & 8'hbb)^
({{8{data_in[93][7]}} & 8'h5d)^
({{8{data_in[94][0]}} & 8'hac)^
({{8{data_in[94][1]}} & 8'h73)^
({{8{data_in[94][2]}} & 8'he6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[94][3]}} & 8'he7)^
({{8{data_in[94][4]}} & 8'he5)^
({{8{data_in[94][5]}} & 8'he1)^
({{8{data_in[94][6]}} & 8'he9)^
({{8{data_in[94][7]}} & 8'hf9)^
({{8{data_in[95][0]}} & 8'h9)^
({{8{data_in[95][1]}} & 8'h12)^
({{8{data_in[95][2]}} & 8'h24)^
({{8{data_in[95][3]}} & 8'h48)^
({{8{data_in[95][4]}} & 8'h90)^
({{8{data_in[95][5]}} & 8'hb)^
({{8{data_in[95][6]}} & 8'h16)^
({{8{data_in[95][7]}} & 8'h2c)^
({{8{data_in[96][0]}} & 8'hf1)^
({{8{data_in[96][1]}} & 8'hc9)^
({{8{data_in[96][2]}} & 8'hb9)^
({{8{data_in[96][3]}} & 8'h59)^
({{8{data_in[96][4]}} & 8'hb2)^
({{8{data_in[96][5]}} & 8'h4f)^
({{8{data_in[96][6]}} & 8'h9e)^
({{8{data_in[96][7]}} & 8'h17)^
({{8{data_in[97][0]}} & 8'h7)^
({{8{data_in[97][1]}} & 8'he)^
({{8{data_in[97][2]}} & 8'h1c)^
({{8{data_in[97][3]}} & 8'h38)^
({{8{data_in[97][4]}} & 8'h70)^
({{8{data_in[97][5]}} & 8'he0)^
({{8{data_in[97][6]}} & 8'heb)^
({{8{data_in[97][7]}} & 8'hfd)^
({{8{data_in[98][0]}} & 8'h3a)^
({{8{data_in[98][1]}} & 8'h74)^
({{8{data_in[98][2]}} & 8'he8)^
({{8{data_in[98][3]}} & 8'hfb)^
({{8{data_in[98][4]}} & 8'hdd)^
({{8{data_in[98][5]}} & 8'h91)^
({{8{data_in[98][6]}} & 8'h9)^
({{8{data_in[98][7]}} & 8'h12)^
({{8{data_in[99][0]}} & 8'h58)^
({{8{data_in[99][1]}} & 8'hb0)^
({{8{data_in[99][2]}} & 8'h4b)^
({{8{data_in[99][3]}} & 8'h96)^
({{8{data_in[99][4]}} & 8'h7)^
({{8{data_in[99][5]}} & 8'he)^
({{8{data_in[99][6]}} & 8'h1c)^
({{8{data_in[99][7]}} & 8'h38)^
({{8{data_in[100][0]}} & 8'hab)^
({{8{data_in[100][1]}} & 8'h7d)^
({{8{data_in[100][2]}} & 8'hfa)^
({{8{data_in[100][3]}} & 8'hdf)^
({{8{data_in[100][4]}} & 8'h95)^
({{8{data_in[100][5]}} & 8'h1)^
({{8{data_in[100][6]}} & 8'h2)^
({{8{data_in[100][7]}} & 8'h4)^
({{8{data_in[101][0]}} & 8'h6b)^
({{8{data_in[101][1]}} & 8'hd6)^
({{8{data_in[101][2]}} & 8'h87)^
({{8{data_in[101][3]}} & 8'h25)^
({{8{data_in[101][4]}} & 8'h4a)^
({{8{data_in[101][5]}} & 8'h94)^
({{8{data_in[101][6]}} & 8'h3)^
({{8{data_in[101][7]}} & 8'h6)^
({{8{data_in[102][0]}} & 8'hd6)^
({{8{data_in[102][1]}} & 8'h87)^
({{8{data_in[102][2]}} & 8'h25)^
({{8{data_in[102][3]}} & 8'h4a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[102][4]}} & 8'h94)^
({{8{data_in[102][5]}} & 8'h3)^
({{8{data_in[102][6]}} & 8'h6)^
({{8{data_in[102][7]}} & 8'hc)^
({{8{data_in[103][0]}} & 8'h58)^
({{8{data_in[103][1]}} & 8'hb0)^
({{8{data_in[103][2]}} & 8'h4b)^
({{8{data_in[103][3]}} & 8'h96)^
({{8{data_in[103][4]}} & 8'h7)^
({{8{data_in[103][5]}} & 8'he)^
({{8{data_in[103][6]}} & 8'h1c)^
({{8{data_in[103][7]}} & 8'h38)^
({{8{data_in[104][0]}} & 8'h90)^
({{8{data_in[104][1]}} & 8'hb)^
({{8{data_in[104][2]}} & 8'h16)^
({{8{data_in[104][3]}} & 8'h2c)^
({{8{data_in[104][4]}} & 8'h58)^
({{8{data_in[104][5]}} & 8'hb0)^
({{8{data_in[104][6]}} & 8'h4b)^
({{8{data_in[104][7]}} & 8'h96)^
({{8{data_in[105][0]}} & 8'hea)^
({{8{data_in[105][1]}} & 8'hff)^
({{8{data_in[105][2]}} & 8'hd5)^
({{8{data_in[105][3]}} & 8'h81)^
({{8{data_in[105][4]}} & 8'h29)^
({{8{data_in[105][5]}} & 8'h52)^
({{8{data_in[105][6]}} & 8'ha4)^
({{8{data_in[105][7]}} & 8'h63)^
({{8{data_in[106][0]}} & 8'h7)^
({{8{data_in[106][1]}} & 8'he)^
({{8{data_in[106][2]}} & 8'h1c)^
({{8{data_in[106][3]}} & 8'h38)^
({{8{data_in[106][4]}} & 8'h70)^
({{8{data_in[106][5]}} & 8'he0)^
({{8{data_in[106][6]}} & 8'heb)^
({{8{data_in[106][7]}} & 8'hfd)^
({{8{data_in[107][0]}} & 8'hd9)^
({{8{data_in[107][1]}} & 8'h99)^
({{8{data_in[107][2]}} & 8'h19)^
({{8{data_in[107][3]}} & 8'h32)^
({{8{data_in[107][4]}} & 8'h64)^
({{8{data_in[107][5]}} & 8'hc8)^
({{8{data_in[107][6]}} & 8'hbb)^
({{8{data_in[107][7]}} & 8'h5d)^
({{8{data_in[108][0]}} & 8'hb9)^
({{8{data_in[108][1]}} & 8'h59)^
({{8{data_in[108][2]}} & 8'hb2)^
({{8{data_in[108][3]}} & 8'h4f)^
({{8{data_in[108][4]}} & 8'h9e)^
({{8{data_in[108][5]}} & 8'h17)^
({{8{data_in[108][6]}} & 8'h2e)^
({{8{data_in[108][7]}} & 8'h5c)^
({{8{data_in[109][0]}} & 8'hce)^
({{8{data_in[109][1]}} & 8'hb7)^
({{8{data_in[109][2]}} & 8'h45)^
({{8{data_in[109][3]}} & 8'h8a)^
({{8{data_in[109][4]}} & 8'h3f)^
({{8{data_in[109][5]}} & 8'h7e)^
({{8{data_in[109][6]}} & 8'hfc)^
({{8{data_in[109][7]}} & 8'hd3)^
({{8{data_in[110][0]}} & 8'h57)^
({{8{data_in[110][1]}} & 8'hae)^
({{8{data_in[110][2]}} & 8'h77)^
({{8{data_in[110][3]}} & 8'hee)^
({{8{data_in[110][4]}} & 8'hf7)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[110][5]}} & 8'hc5)^
({{8{data_in[110][6]}} & 8'ha1)^
({{8{data_in[110][7]}} & 8'h69)^
({{8{data_in[111][0]}} & 8'hf5)^
({{8{data_in[111][1]}} & 8'hc1)^
({{8{data_in[111][2]}} & 8'ha9)^
({{8{data_in[111][3]}} & 8'h79)^
({{8{data_in[111][4]}} & 8'hf2)^
({{8{data_in[111][5]}} & 8'hcf)^
({{8{data_in[111][6]}} & 8'hb5)^
({{8{data_in[111][7]}} & 8'h41)^
({{8{data_in[112][0]}} & 8'h2a)^
({{8{data_in[112][1]}} & 8'h54)^
({{8{data_in[112][2]}} & 8'ha8)^
({{8{data_in[112][3]}} & 8'h7b)^
({{8{data_in[112][4]}} & 8'hf6)^
({{8{data_in[112][5]}} & 8'hc7)^
({{8{data_in[112][6]}} & 8'ha5)^
({{8{data_in[112][7]}} & 8'h61)^
({{8{data_in[113][0]}} & 8'hdf)^
({{8{data_in[113][1]}} & 8'h95)^
({{8{data_in[113][2]}} & 8'h1)^
({{8{data_in[113][3]}} & 8'h2)^
({{8{data_in[113][4]}} & 8'h4)^
({{8{data_in[113][5]}} & 8'h8)^
({{8{data_in[113][6]}} & 8'h10)^
({{8{data_in[113][7]}} & 8'h20)^
({{8{data_in[114][0]}} & 8'h86)^
({{8{data_in[114][1]}} & 8'h27)^
({{8{data_in[114][2]}} & 8'h4e)^
({{8{data_in[114][3]}} & 8'h9c)^
({{8{data_in[114][4]}} & 8'h13)^
({{8{data_in[114][5]}} & 8'h26)^
({{8{data_in[114][6]}} & 8'h4c)^
({{8{data_in[114][7]}} & 8'h98)^
({{8{data_in[115][0]}} & 8'h4c)^
({{8{data_in[115][1]}} & 8'h98)^
({{8{data_in[115][2]}} & 8'h1b)^
({{8{data_in[115][3]}} & 8'h36)^
({{8{data_in[115][4]}} & 8'h6c)^
({{8{data_in[115][5]}} & 8'hd8)^
({{8{data_in[115][6]}} & 8'h9b)^
({{8{data_in[115][7]}} & 8'h1d)^
({{8{data_in[116][0]}} & 8'h48)^
({{8{data_in[116][1]}} & 8'h90)^
({{8{data_in[116][2]}} & 8'hb)^
({{8{data_in[116][3]}} & 8'h16)^
({{8{data_in[116][4]}} & 8'h2c)^
({{8{data_in[116][5]}} & 8'h58)^
({{8{data_in[116][6]}} & 8'hb0)^
({{8{data_in[116][7]}} & 8'h4b)^
({{8{data_in[117][0]}} & 8'h63)^
({{8{data_in[117][1]}} & 8'hc6)^
({{8{data_in[117][2]}} & 8'ha7)^
({{8{data_in[117][3]}} & 8'h65)^
({{8{data_in[117][4]}} & 8'hca)^
({{8{data_in[117][5]}} & 8'hbf)^
({{8{data_in[117][6]}} & 8'h55)^
({{8{data_in[117][7]}} & 8'haa)^
({{8{data_in[118][0]}} & 8'h57)^
({{8{data_in[118][1]}} & 8'hae)^
({{8{data_in[118][2]}} & 8'h77)^
({{8{data_in[118][3]}} & 8'hee)^
({{8{data_in[118][4]}} & 8'hf7)^
({{8{data_in[118][5]}} & 8'hc5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[118][6]}} & 8'ha1)^
({{8{data_in[118][7]}} & 8'h69)^
({{8{data_in[119][0]}} & 8'h11)^
({{8{data_in[119][1]}} & 8'h22)^
({{8{data_in[119][2]}} & 8'h44)^
({{8{data_in[119][3]}} & 8'h88)^
({{8{data_in[119][4]}} & 8'h3b)^
({{8{data_in[119][5]}} & 8'h76)^
({{8{data_in[119][6]}} & 8'hec)^
({{8{data_in[119][7]}} & 8'hf3)^
({{8{data_in[120][0]}} & 8'h25)^
({{8{data_in[120][1]}} & 8'h4a)^
({{8{data_in[120][2]}} & 8'h94)^
({{8{data_in[120][3]}} & 8'h3)^
({{8{data_in[120][4]}} & 8'h6)^
({{8{data_in[120][5]}} & 8'hc)^
({{8{data_in[120][6]}} & 8'h18)^
({{8{data_in[120][7]}} & 8'h30)^
({{8{data_in[121][0]}} & 8'h8a)^
({{8{data_in[121][1]}} & 8'h3f)^
({{8{data_in[121][2]}} & 8'h7e)^
({{8{data_in[121][3]}} & 8'hfc)^
({{8{data_in[121][4]}} & 8'hd3)^
({{8{data_in[121][5]}} & 8'h8d)^
({{8{data_in[121][6]}} & 8'h31)^
({{8{data_in[121][7]}} & 8'h62)^
({{8{data_in[122][0]}} & 8'h91)^
({{8{data_in[122][1]}} & 8'h9)^
({{8{data_in[122][2]}} & 8'h12)^
({{8{data_in[122][3]}} & 8'h24)^
({{8{data_in[122][4]}} & 8'h48)^
({{8{data_in[122][5]}} & 8'h90)^
({{8{data_in[122][6]}} & 8'hb)^
({{8{data_in[122][7]}} & 8'h16)^
({{8{data_in[123][0]}} & 8'hd6)^
({{8{data_in[123][1]}} & 8'h87)^
({{8{data_in[123][2]}} & 8'h25)^
({{8{data_in[123][3]}} & 8'h4a)^
({{8{data_in[123][4]}} & 8'h94)^
({{8{data_in[123][5]}} & 8'h3)^
({{8{data_in[123][6]}} & 8'h6)^
({{8{data_in[123][7]}} & 8'hc)^
({{8{data_in[124][0]}} & 8'h46)^
({{8{data_in[124][1]}} & 8'h8c)^
({{8{data_in[124][2]}} & 8'h33)^
({{8{data_in[124][3]}} & 8'h66)^
({{8{data_in[124][4]}} & 8'hcc)^
({{8{data_in[124][5]}} & 8'hb3)^
({{8{data_in[124][6]}} & 8'h4d)^
({{8{data_in[124][7]}} & 8'h9a)^
({{8{data_in[125][0]}} & 8'ha2)^
({{8{data_in[125][1]}} & 8'h6f)^
({{8{data_in[125][2]}} & 8'hde)^
({{8{data_in[125][3]}} & 8'h97)^
({{8{data_in[125][4]}} & 8'h5)^
({{8{data_in[125][5]}} & 8'ha)^
({{8{data_in[125][6]}} & 8'h14)^
({{8{data_in[125][7]}} & 8'h28)^
({{8{data_in[126][0]}} & 8'h45)^
({{8{data_in[126][1]}} & 8'h8a)^
({{8{data_in[126][2]}} & 8'h3f)^
({{8{data_in[126][3]}} & 8'h7e)^
({{8{data_in[126][4]}} & 8'hfc)^
({{8{data_in[126][5]}} & 8'hd3)^
({{8{data_in[126][6]}} & 8'h8d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[126][7]}} & 8'h31)^
({{8{data_in[127][0]}} & 8'hd)^
({{8{data_in[127][1]}} & 8'h1a)^
({{8{data_in[127][2]}} & 8'h34)^
({{8{data_in[127][3]}} & 8'h68)^
({{8{data_in[127][4]}} & 8'hd0)^
({{8{data_in[127][5]}} & 8'h8b)^
({{8{data_in[127][6]}} & 8'h3d)^
({{8{data_in[127][7]}} & 8'h7a)^
({{8{data_in[128][0]}} & 8'hb0)^
({{8{data_in[128][1]}} & 8'h4b)^
({{8{data_in[128][2]}} & 8'h96)^
({{8{data_in[128][3]}} & 8'h7)^
({{8{data_in[128][4]}} & 8'he)^
({{8{data_in[128][5]}} & 8'h1c)^
({{8{data_in[128][6]}} & 8'h38)^
({{8{data_in[128][7]}} & 8'h70)^
({{8{data_in[129][0]}} & 8'h7d)^
({{8{data_in[129][1]}} & 8'hfa)^
({{8{data_in[129][2]}} & 8'hdf)^
({{8{data_in[129][3]}} & 8'h95)^
({{8{data_in[129][4]}} & 8'h1)^
({{8{data_in[129][5]}} & 8'h2)^
({{8{data_in[129][6]}} & 8'h4)^
({{8{data_in[129][7]}} & 8'h8)^
({{8{data_in[130][0]}} & 8'h2e)^
({{8{data_in[130][1]}} & 8'h5c)^
({{8{data_in[130][2]}} & 8'hb8)^
({{8{data_in[130][3]}} & 8'h5b)^
({{8{data_in[130][4]}} & 8'hb6)^
({{8{data_in[130][5]}} & 8'h47)^
({{8{data_in[130][6]}} & 8'h8e)^
({{8{data_in[130][7]}} & 8'h37)^
({{8{data_in[131][0]}} & 8'hcc)^
({{8{data_in[131][1]}} & 8'hb3)^
({{8{data_in[131][2]}} & 8'h4d)^
({{8{data_in[131][3]}} & 8'h9a)^
({{8{data_in[131][4]}} & 8'h1f)^
({{8{data_in[131][5]}} & 8'h3e)^
({{8{data_in[131][6]}} & 8'h7c)^
({{8{data_in[131][7]}} & 8'hf8)^
({{8{data_in[132][0]}} & 8'h2b)^
({{8{data_in[132][1]}} & 8'h56)^
({{8{data_in[132][2]}} & 8'hac)^
({{8{data_in[132][3]}} & 8'h73)^
({{8{data_in[132][4]}} & 8'he6)^
({{8{data_in[132][5]}} & 8'he7)^
({{8{data_in[132][6]}} & 8'he5)^
({{8{data_in[132][7]}} & 8'he1)^
({{8{data_in[133][0]}} & 8'hd4)^
({{8{data_in[133][1]}} & 8'h83)^
({{8{data_in[133][2]}} & 8'h2d)^
({{8{data_in[133][3]}} & 8'h5a)^
({{8{data_in[133][4]}} & 8'hb4)^
({{8{data_in[133][5]}} & 8'h43)^
({{8{data_in[133][6]}} & 8'h86)^
({{8{data_in[133][7]}} & 8'h27)^
({{8{data_in[134][0]}} & 8'h24)^
({{8{data_in[134][1]}} & 8'h48)^
({{8{data_in[134][2]}} & 8'h90)^
({{8{data_in[134][3]}} & 8'hb)^
({{8{data_in[134][4]}} & 8'h16)^
({{8{data_in[134][5]}} & 8'h2c)^
({{8{data_in[134][6]}} & 8'h58)^
({{8{data_in[134][7]}} & 8'hb0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[135][0]}} & 8'h7a)^
({{8{data_in[135][1]}} & 8'hf4)^
({{8{data_in[135][2]}} & 8'hc3)^
({{8{data_in[135][3]}} & 8'had)^
({{8{data_in[135][4]}} & 8'h71)^
({{8{data_in[135][5]}} & 8'he2)^
({{8{data_in[135][6]}} & 8'hef)^
({{8{data_in[135][7]}} & 8'hf5)^
({{8{data_in[136][0]}} & 8'h5a)^
({{8{data_in[136][1]}} & 8'hb4)^
({{8{data_in[136][2]}} & 8'h43)^
({{8{data_in[136][3]}} & 8'h86)^
({{8{data_in[136][4]}} & 8'h27)^
({{8{data_in[136][5]}} & 8'h4e)^
({{8{data_in[136][6]}} & 8'h9c)^
({{8{data_in[136][7]}} & 8'h13)^
({{8{data_in[137][0]}} & 8'he5)^
({{8{data_in[137][1]}} & 8'he1)^
({{8{data_in[137][2]}} & 8'he9)^
({{8{data_in[137][3]}} & 8'hf9)^
({{8{data_in[137][4]}} & 8'hd9)^
({{8{data_in[137][5]}} & 8'h99)^
({{8{data_in[137][6]}} & 8'h19)^
({{8{data_in[137][7]}} & 8'h32)^
({{8{data_in[138][0]}} & 8'hc7)^
({{8{data_in[138][1]}} & 8'ha5)^
({{8{data_in[138][2]}} & 8'h61)^
({{8{data_in[138][3]}} & 8'hc2)^
({{8{data_in[138][4]}} & 8'haf)^
({{8{data_in[138][5]}} & 8'h75)^
({{8{data_in[138][6]}} & 8'hea)^
({{8{data_in[138][7]}} & 8'hff)^
({{8{data_in[139][0]}} & 8'hf0)^
({{8{data_in[139][1]}} & 8'hcb)^
({{8{data_in[139][2]}} & 8'hbd)^
({{8{data_in[139][3]}} & 8'h51)^
({{8{data_in[139][4]}} & 8'ha2)^
({{8{data_in[139][5]}} & 8'h6f)^
({{8{data_in[139][6]}} & 8'hde)^
({{8{data_in[139][7]}} & 8'h97)^
({{8{data_in[140][0]}} & 8'h65)^
({{8{data_in[140][1]}} & 8'hca)^
({{8{data_in[140][2]}} & 8'hbf)^
({{8{data_in[140][3]}} & 8'h55)^
({{8{data_in[140][4]}} & 8'haa)^
({{8{data_in[140][5]}} & 8'h7f)^
({{8{data_in[140][6]}} & 8'hfe)^
({{8{data_in[140][7]}} & 8'hd7)^
({{8{data_in[141][0]}} & 8'h6a)^
({{8{data_in[141][1]}} & 8'hd4)^
({{8{data_in[141][2]}} & 8'h83)^
({{8{data_in[141][3]}} & 8'h2d)^
({{8{data_in[141][4]}} & 8'h5a)^
({{8{data_in[141][5]}} & 8'hb4)^
({{8{data_in[141][6]}} & 8'h43)^
({{8{data_in[141][7]}} & 8'h86)^
({{8{data_in[142][0]}} & 8'hb6)^
({{8{data_in[142][1]}} & 8'h47)^
({{8{data_in[142][2]}} & 8'h8e)^
({{8{data_in[142][3]}} & 8'h37)^
({{8{data_in[142][4]}} & 8'h6e)^
({{8{data_in[142][5]}} & 8'hdc)^
({{8{data_in[142][6]}} & 8'h93)^
({{8{data_in[142][7]}} & 8'hd)^
({{8{data_in[143][0]}} & 8'hee)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[143][1]}} & 8'hf7)^
({{8{data_in[143][2]}} & 8'hc5)^
({{8{data_in[143][3]}} & 8'ha1)^
({{8{data_in[143][4]}} & 8'h69)^
({{8{data_in[143][5]}} & 8'hd2)^
({{8{data_in[143][6]}} & 8'h8f)^
({{8{data_in[143][7]}} & 8'h35)^
({{8{data_in[144][0]}} & 8'h97)^
({{8{data_in[144][1]}} & 8'h5)^
({{8{data_in[144][2]}} & 8'ha)^
({{8{data_in[144][3]}} & 8'h14)^
({{8{data_in[144][4]}} & 8'h28)^
({{8{data_in[144][5]}} & 8'h50)^
({{8{data_in[144][6]}} & 8'ha0)^
({{8{data_in[144][7]}} & 8'h6b)^
({{8{data_in[145][0]}} & 8'h12)^
({{8{data_in[145][1]}} & 8'h24)^
({{8{data_in[145][2]}} & 8'h48)^
({{8{data_in[145][3]}} & 8'h90)^
({{8{data_in[145][4]}} & 8'hb)^
({{8{data_in[145][5]}} & 8'h16)^
({{8{data_in[145][6]}} & 8'h2c)^
({{8{data_in[145][7]}} & 8'h58)^
({{8{data_in[146][0]}} & 8'h18)^
({{8{data_in[146][1]}} & 8'h30)^
({{8{data_in[146][2]}} & 8'h60)^
({{8{data_in[146][3]}} & 8'hc0)^
({{8{data_in[146][4]}} & 8'hab)^
({{8{data_in[146][5]}} & 8'h7d)^
({{8{data_in[146][6]}} & 8'hfa)^
({{8{data_in[146][7]}} & 8'hdf)^
({{8{data_in[147][0]}} & 8'hb9)^
({{8{data_in[147][1]}} & 8'h59)^
({{8{data_in[147][2]}} & 8'hb2)^
({{8{data_in[147][3]}} & 8'h4f)^
({{8{data_in[147][4]}} & 8'h9e)^
({{8{data_in[147][5]}} & 8'h17)^
({{8{data_in[147][6]}} & 8'h2e)^
({{8{data_in[147][7]}} & 8'h5c)^
({{8{data_in[148][0]}} & 8'hf7)^
({{8{data_in[148][1]}} & 8'hc5)^
({{8{data_in[148][2]}} & 8'ha1)^
({{8{data_in[148][3]}} & 8'h69)^
({{8{data_in[148][4]}} & 8'hd2)^
({{8{data_in[148][5]}} & 8'h8f)^
({{8{data_in[148][6]}} & 8'h35)^
({{8{data_in[148][7]}} & 8'h6a)^
({{8{data_in[149][0]}} & 8'h6a)^
({{8{data_in[149][1]}} & 8'hd4)^
({{8{data_in[149][2]}} & 8'h83)^
({{8{data_in[149][3]}} & 8'h2d)^
({{8{data_in[149][4]}} & 8'h5a)^
({{8{data_in[149][5]}} & 8'hb4)^
({{8{data_in[149][6]}} & 8'h43)^
({{8{data_in[149][7]}} & 8'h86)^
({{8{data_in[150][0]}} & 8'h14)^
({{8{data_in[150][1]}} & 8'h28)^
({{8{data_in[150][2]}} & 8'h50)^
({{8{data_in[150][3]}} & 8'ha0)^
({{8{data_in[150][4]}} & 8'h6b)^
({{8{data_in[150][5]}} & 8'hd6)^
({{8{data_in[150][6]}} & 8'h87)^
({{8{data_in[150][7]}} & 8'h25)^
({{8{data_in[151][0]}} & 8'hc8)^
({{8{data_in[151][1]}} & 8'hbb)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[151][2]}} & 8'h5d)^
({{8{data_in[151][3]}} & 8'hba)^
({{8{data_in[151][4]}} & 8'h5f)^
({{8{data_in[151][5]}} & 8'hbe)^
({{8{data_in[151][6]}} & 8'h57)^
({{8{data_in[151][7]}} & 8'hae)^
({{8{data_in[152][0]}} & 8'h8a)^
({{8{data_in[152][1]}} & 8'h3f)^
({{8{data_in[152][2]}} & 8'h7e)^
({{8{data_in[152][3]}} & 8'hfc)^
({{8{data_in[152][4]}} & 8'hd3)^
({{8{data_in[152][5]}} & 8'h8d)^
({{8{data_in[152][6]}} & 8'h31)^
({{8{data_in[152][7]}} & 8'h62)^
({{8{data_in[153][0]}} & 8'ha8)^
({{8{data_in[153][1]}} & 8'h7b)^
({{8{data_in[153][2]}} & 8'hf6)^
({{8{data_in[153][3]}} & 8'hc7)^
({{8{data_in[153][4]}} & 8'ha5)^
({{8{data_in[153][5]}} & 8'h61)^
({{8{data_in[153][6]}} & 8'hc2)^
({{8{data_in[153][7]}} & 8'haf)^
({{8{data_in[154][0]}} & 8'h4f)^
({{8{data_in[154][1]}} & 8'h9e)^
({{8{data_in[154][2]}} & 8'h17)^
({{8{data_in[154][3]}} & 8'h2e)^
({{8{data_in[154][4]}} & 8'h5c)^
({{8{data_in[154][5]}} & 8'hb8)^
({{8{data_in[154][6]}} & 8'h5b)^
({{8{data_in[154][7]}} & 8'hb6)^
({{8{data_in[155][0]}} & 8'h27)^
({{8{data_in[155][1]}} & 8'h4e)^
({{8{data_in[155][2]}} & 8'h9c)^
({{8{data_in[155][3]}} & 8'h13)^
({{8{data_in[155][4]}} & 8'h26)^
({{8{data_in[155][5]}} & 8'h4c)^
({{8{data_in[155][6]}} & 8'h98)^
({{8{data_in[155][7]}} & 8'h1b)^
({{8{data_in[156][0]}} & 8'h97)^
({{8{data_in[156][1]}} & 8'h5)^
({{8{data_in[156][2]}} & 8'ha)^
({{8{data_in[156][3]}} & 8'h14)^
({{8{data_in[156][4]}} & 8'h28)^
({{8{data_in[156][5]}} & 8'h50)^
({{8{data_in[156][6]}} & 8'ha0)^
({{8{data_in[156][7]}} & 8'h6b)^
({{8{data_in[157][0]}} & 8'h34)^
({{8{data_in[157][1]}} & 8'h68)^
({{8{data_in[157][2]}} & 8'hd0)^
({{8{data_in[157][3]}} & 8'h8b)^
({{8{data_in[157][4]}} & 8'h3d)^
({{8{data_in[157][5]}} & 8'h7a)^
({{8{data_in[157][6]}} & 8'hf4)^
({{8{data_in[157][7]}} & 8'hc3)^
({{8{data_in[158][0]}} & 8'hf2)^
({{8{data_in[158][1]}} & 8'hcf)^
({{8{data_in[158][2]}} & 8'hb5)^
({{8{data_in[158][3]}} & 8'h41)^
({{8{data_in[158][4]}} & 8'h82)^
({{8{data_in[158][5]}} & 8'h2f)^
({{8{data_in[158][6]}} & 8'h5e)^
({{8{data_in[158][7]}} & 8'hbc)^
({{8{data_in[159][0]}} & 8'h96)^
({{8{data_in[159][1]}} & 8'h7)^
({{8{data_in[159][2]}} & 8'he})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[159][3]}} & 8'h1c)^
({{8{data_in[159][4]}} & 8'h38)^
({{8{data_in[159][5]}} & 8'h70)^
({{8{data_in[159][6]}} & 8'he0)^
({{8{data_in[159][7]}} & 8'heb)^
({{8{data_in[160][0]}} & 8'h71)^
({{8{data_in[160][1]}} & 8'he2)^
({{8{data_in[160][2]}} & 8'hef)^
({{8{data_in[160][3]}} & 8'hf5)^
({{8{data_in[160][4]}} & 8'hc1)^
({{8{data_in[160][5]}} & 8'ha9)^
({{8{data_in[160][6]}} & 8'h79)^
({{8{data_in[160][7]}} & 8'hf2)^
({{8{data_in[161][0]}} & 8'h7)^
({{8{data_in[161][1]}} & 8'he)^
({{8{data_in[161][2]}} & 8'h1c)^
({{8{data_in[161][3]}} & 8'h38)^
({{8{data_in[161][4]}} & 8'h70)^
({{8{data_in[161][5]}} & 8'he0)^
({{8{data_in[161][6]}} & 8'heb)^
({{8{data_in[161][7]}} & 8'bfd)^
({{8{data_in[162][0]}} & 8'h94)^
({{8{data_in[162][1]}} & 8'h3)^
({{8{data_in[162][2]}} & 8'h6)^
({{8{data_in[162][3]}} & 8'hc)^
({{8{data_in[162][4]}} & 8'h18)^
({{8{data_in[162][5]}} & 8'h30)^
({{8{data_in[162][6]}} & 8'h60)^
({{8{data_in[162][7]}} & 8'hc0)^
({{8{data_in[163][0]}} & 8'h10)^
({{8{data_in[163][1]}} & 8'h20)^
({{8{data_in[163][2]}} & 8'h40)^
({{8{data_in[163][3]}} & 8'h80)^
({{8{data_in[163][4]}} & 8'h2b)^
({{8{data_in[163][5]}} & 8'h56)^
({{8{data_in[163][6]}} & 8'hac)^
({{8{data_in[163][7]}} & 8'h73)^
({{8{data_in[164][0]}} & 8'h7a)^
({{8{data_in[164][1]}} & 8'hf4)^
({{8{data_in[164][2]}} & 8'hc3)^
({{8{data_in[164][3]}} & 8'had)^
({{8{data_in[164][4]}} & 8'h71)^
({{8{data_in[164][5]}} & 8'he2)^
({{8{data_in[164][6]}} & 8'hef)^
({{8{data_in[164][7]}} & 8'hf5)^
({{8{data_in[165][0]}} & 8'hf)^
({{8{data_in[165][1]}} & 8'h1e)^
({{8{data_in[165][2]}} & 8'h3c)^
({{8{data_in[165][3]}} & 8'h78)^
({{8{data_in[165][4]}} & 8'hf0)^
({{8{data_in[165][5]}} & 8'hcb)^
({{8{data_in[165][6]}} & 8'hbd)^
({{8{data_in[165][7]}} & 8'h51)^
({{8{data_in[166][0]}} & 8'h20)^
({{8{data_in[166][1]}} & 8'h40)^
({{8{data_in[166][2]}} & 8'h80)^
({{8{data_in[166][3]}} & 8'h2b)^
({{8{data_in[166][4]}} & 8'h56)^
({{8{data_in[166][5]}} & 8'hac)^
({{8{data_in[166][6]}} & 8'h73)^
({{8{data_in[166][7]}} & 8'he6)^
({{8{data_in[167][0]}} & 8'h11)^
({{8{data_in[167][1]}} & 8'h22)^
({{8{data_in[167][2]}} & 8'h44)^
({{8{data_in[167][3]}} & 8'h88)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[167][4]}} & 8'h3b)^
({{8{data_in[167][5]}} & 8'h76)^
({{8{data_in[167][6]}} & 8'hec)^
({{8{data_in[167][7]}} & 8'hf3)^
({{8{data_in[168][0]}} & 8'h29)^
({{8{data_in[168][1]}} & 8'h52)^
({{8{data_in[168][2]}} & 8'ha4)^
({{8{data_in[168][3]}} & 8'h63)^
({{8{data_in[168][4]}} & 8'hc6)^
({{8{data_in[168][5]}} & 8'ha7)^
({{8{data_in[168][6]}} & 8'h65)^
({{8{data_in[168][7]}} & 8'hca)^
({{8{data_in[169][0]}} & 8'h7c)^
({{8{data_in[169][1]}} & 8'hf8)^
({{8{data_in[169][2]}} & 8'hdb)^
({{8{data_in[169][3]}} & 8'h9d)^
({{8{data_in[169][4]}} & 8'h11)^
({{8{data_in[169][5]}} & 8'h22)^
({{8{data_in[169][6]}} & 8'h44)^
({{8{data_in[169][7]}} & 8'h88)^
({{8{data_in[170][0]}} & 8'h25)^
({{8{data_in[170][1]}} & 8'h4a)^
({{8{data_in[170][2]}} & 8'h94)^
({{8{data_in[170][3]}} & 8'h3)^
({{8{data_in[170][4]}} & 8'h6)^
({{8{data_in[170][5]}} & 8'hc)^
({{8{data_in[170][6]}} & 8'h18)^
({{8{data_in[170][7]}} & 8'h30)^
({{8{data_in[171][0]}} & 8'h65)^
({{8{data_in[171][1]}} & 8'hca)^
({{8{data_in[171][2]}} & 8'hbf)^
({{8{data_in[171][3]}} & 8'h55)^
({{8{data_in[171][4]}} & 8'haa)^
({{8{data_in[171][5]}} & 8'h7f)^
({{8{data_in[171][6]}} & 8'hfe)^
({{8{data_in[171][7]}} & 8'hd7)^
({{8{data_in[172][0]}} & 8'h1e)^
({{8{data_in[172][1]}} & 8'h3c)^
({{8{data_in[172][2]}} & 8'h78)^
({{8{data_in[172][3]}} & 8'hf0)^
({{8{data_in[172][4]}} & 8'hcb)^
({{8{data_in[172][5]}} & 8'hbd)^
({{8{data_in[172][6]}} & 8'h51)^
({{8{data_in[172][7]}} & 8'ha2)^
({{8{data_in[173][0]}} & 8'h54)^
({{8{data_in[173][1]}} & 8'ha8)^
({{8{data_in[173][2]}} & 8'h7b)^
({{8{data_in[173][3]}} & 8'hf6)^
({{8{data_in[173][4]}} & 8'hc7)^
({{8{data_in[173][5]}} & 8'ha5)^
({{8{data_in[173][6]}} & 8'h61)^
({{8{data_in[173][7]}} & 8'hc2)^
({{8{data_in[174][0]}} & 8'h8b)^
({{8{data_in[174][1]}} & 8'h3d)^
({{8{data_in[174][2]}} & 8'h7a)^
({{8{data_in[174][3]}} & 8'hf4)^
({{8{data_in[174][4]}} & 8'hc3)^
({{8{data_in[174][5]}} & 8'had)^
({{8{data_in[174][6]}} & 8'h71)^
({{8{data_in[174][7]}} & 8'he2)^
({{8{data_in[175][0]}} & 8'h21)^
({{8{data_in[175][1]}} & 8'h42)^
({{8{data_in[175][2]}} & 8'h84)^
({{8{data_in[175][3]}} & 8'h23)^
({{8{data_in[175][4]}} & 8'h46)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[175][5]}} & 8'h8c)^
({{8{data_in[175][6]}} & 8'h33)^
({{8{data_in[175][7]}} & 8'h66)^
({{8{data_in[176][0]}} & 8'hdd)^
({{8{data_in[176][1]}} & 8'h91)^
({{8{data_in[176][2]}} & 8'h9)^
({{8{data_in[176][3]}} & 8'h12)^
({{8{data_in[176][4]}} & 8'h24)^
({{8{data_in[176][5]}} & 8'h48)^
({{8{data_in[176][6]}} & 8'h90)^
({{8{data_in[176][7]}} & 8'hb)^
({{8{data_in[177][0]}} & 8'h13)^
({{8{data_in[177][1]}} & 8'h26)^
({{8{data_in[177][2]}} & 8'h4c)^
({{8{data_in[177][3]}} & 8'h98)^
({{8{data_in[177][4]}} & 8'h1b)^
({{8{data_in[177][5]}} & 8'h36)^
({{8{data_in[177][6]}} & 8'h6c)^
({{8{data_in[177][7]}} & 8'hd8)^
({{8{data_in[178][0]}} & 8'h10)^
({{8{data_in[178][1]}} & 8'h20)^
({{8{data_in[178][2]}} & 8'h40)^
({{8{data_in[178][3]}} & 8'h80)^
({{8{data_in[178][4]}} & 8'h2b)^
({{8{data_in[178][5]}} & 8'h56)^
({{8{data_in[178][6]}} & 8'hac)^
({{8{data_in[178][7]}} & 8'h73)^
({{8{data_in[179][0]}} & 8'h2)^
({{8{data_in[179][1]}} & 8'h4)^
({{8{data_in[179][2]}} & 8'h8)^
({{8{data_in[179][3]}} & 8'h10)^
({{8{data_in[179][4]}} & 8'h20)^
({{8{data_in[179][5]}} & 8'h40)^
({{8{data_in[179][6]}} & 8'h80)^
({{8{data_in[179][7]}} & 8'h2b)^
({{8{data_in[180][0]}} & 8'h53)^
({{8{data_in[180][1]}} & 8'ha6)^
({{8{data_in[180][2]}} & 8'h67)^
({{8{data_in[180][3]}} & 8'hce)^
({{8{data_in[180][4]}} & 8'hb7)^
({{8{data_in[180][5]}} & 8'h45)^
({{8{data_in[180][6]}} & 8'h8a)^
({{8{data_in[180][7]}} & 8'h3f)^
({{8{data_in[181][0]}} & 8'h24)^
({{8{data_in[181][1]}} & 8'h48)^
({{8{data_in[181][2]}} & 8'h90)^
({{8{data_in[181][3]}} & 8'hb)^
({{8{data_in[181][4]}} & 8'h16)^
({{8{data_in[181][5]}} & 8'h2c)^
({{8{data_in[181][6]}} & 8'h58)^
({{8{data_in[181][7]}} & 8'hb0)^
({{8{data_in[182][0]}} & 8'h76)^
({{8{data_in[182][1]}} & 8'hec)^
({{8{data_in[182][2]}} & 8'hf3)^
({{8{data_in[182][3]}} & 8'hcd)^
({{8{data_in[182][4]}} & 8'hb1)^
({{8{data_in[182][5]}} & 8'h49)^
({{8{data_in[182][6]}} & 8'h92)^
({{8{data_in[182][7]}} & 8'hf)^
({{8{data_in[183][0]}} & 8'h41)^
({{8{data_in[183][1]}} & 8'h82)^
({{8{data_in[183][2]}} & 8'h2f)^
({{8{data_in[183][3]}} & 8'h5e)^
({{8{data_in[183][4]}} & 8'hbc)^
({{8{data_in[183][5]}} & 8'h53)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[183][6]}} & 8'ha6)^
({{8{data_in[183][7]}} & 8'h67)^
({{8{data_in[184][0]}} & 8'hbf)^
({{8{data_in[184][1]}} & 8'h55)^
({{8{data_in[184][2]}} & 8'haa)^
({{8{data_in[184][3]}} & 8'h7f)^
({{8{data_in[184][4]}} & 8'hfe)^
({{8{data_in[184][5]}} & 8'hd7)^
({{8{data_in[184][6]}} & 8'h85)^
({{8{data_in[184][7]}} & 8'h21)^
({{8{data_in[185][0]}} & 8'h55)^
({{8{data_in[185][1]}} & 8'haa)^
({{8{data_in[185][2]}} & 8'h7f)^
({{8{data_in[185][3]}} & 8'hfe)^
({{8{data_in[185][4]}} & 8'hd7)^
({{8{data_in[185][5]}} & 8'h85)^
({{8{data_in[185][6]}} & 8'h21)^
({{8{data_in[185][7]}} & 8'h42)^
({{8{data_in[186][0]}} & 8'h75)^
({{8{data_in[186][1]}} & 8'hea)^
({{8{data_in[186][2]}} & 8'hff)^
({{8{data_in[186][3]}} & 8'hd5)^
({{8{data_in[186][4]}} & 8'h81)^
({{8{data_in[186][5]}} & 8'h29)^
({{8{data_in[186][6]}} & 8'h52)^
({{8{data_in[186][7]}} & 8'ha4)^
({{8{data_in[187][0]}} & 8'h7e)^
({{8{data_in[187][1]}} & 8'hfc)^
({{8{data_in[187][2]}} & 8'hd3)^
({{8{data_in[187][3]}} & 8'h8d)^
({{8{data_in[187][4]}} & 8'h31)^
({{8{data_in[187][5]}} & 8'h62)^
({{8{data_in[187][6]}} & 8'hc4)^
({{8{data_in[187][7]}} & 8'ha3)^
({{8{data_in[188][0]}} & 8'hc6)^
({{8{data_in[188][1]}} & 8'ha7)^
({{8{data_in[188][2]}} & 8'h65)^
({{8{data_in[188][3]}} & 8'hca)^
({{8{data_in[188][4]}} & 8'hbf)^
({{8{data_in[188][5]}} & 8'h55)^
({{8{data_in[188][6]}} & 8'haa)^
({{8{data_in[188][7]}} & 8'h7f)^
({{8{data_in[189][0]}} & 8'hbe)^
({{8{data_in[189][1]}} & 8'h57)^
({{8{data_in[189][2]}} & 8'hae)^
({{8{data_in[189][3]}} & 8'h77)^
({{8{data_in[189][4]}} & 8'hee)^
({{8{data_in[189][5]}} & 8'hf7)^
({{8{data_in[189][6]}} & 8'hc5)^
({{8{data_in[189][7]}} & 8'ha1)^
({{8{data_in[190][0]}} & 8'h6a)^
({{8{data_in[190][1]}} & 8'hd4)^
({{8{data_in[190][2]}} & 8'h83)^
({{8{data_in[190][3]}} & 8'h2d)^
({{8{data_in[190][4]}} & 8'h5a)^
({{8{data_in[190][5]}} & 8'hb4)^
({{8{data_in[190][6]}} & 8'h43)^
({{8{data_in[190][7]}} & 8'h86)^
({{8{data_in[191][0]}} & 8'h9f)^
({{8{data_in[191][1]}} & 8'h15)^
({{8{data_in[191][2]}} & 8'h2a)^
({{8{data_in[191][3]}} & 8'h54)^
({{8{data_in[191][4]}} & 8'ha8)^
({{8{data_in[191][5]}} & 8'h7b)^
({{8{data_in[191][6]}} & 8'hf6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[191][7]}} & 8'hc7)^
({{8{data_in[192][0]}} & 8'haa)^
({{8{data_in[192][1]}} & 8'h7f)^
({{8{data_in[192][2]}} & 8'hfe)^
({{8{data_in[192][3]}} & 8'hd7)^
({{8{data_in[192][4]}} & 8'h85)^
({{8{data_in[192][5]}} & 8'h21)^
({{8{data_in[192][6]}} & 8'h42)^
({{8{data_in[192][7]}} & 8'h84)^
({{8{data_in[193][0]}} & 8'h3e)^
({{8{data_in[193][1]}} & 8'h7c)^
({{8{data_in[193][2]}} & 8'hf8)^
({{8{data_in[193][3]}} & 8'hdb)^
({{8{data_in[193][4]}} & 8'h9d)^
({{8{data_in[193][5]}} & 8'h11)^
({{8{data_in[193][6]}} & 8'h22)^
({{8{data_in[193][7]}} & 8'h44)^
({{8{data_in[194][0]}} & 8'h6a)^
({{8{data_in[194][1]}} & 8'hd4)^
({{8{data_in[194][2]}} & 8'h83)^
({{8{data_in[194][3]}} & 8'h2d)^
({{8{data_in[194][4]}} & 8'h5a)^
({{8{data_in[194][5]}} & 8'hb4)^
({{8{data_in[194][6]}} & 8'h43)^
({{8{data_in[194][7]}} & 8'h86)^
({{8{data_in[195][0]}} & 8'h45)^
({{8{data_in[195][1]}} & 8'h8a)^
({{8{data_in[195][2]}} & 8'h3f)^
({{8{data_in[195][3]}} & 8'h7e)^
({{8{data_in[195][4]}} & 8'hfc)^
({{8{data_in[195][5]}} & 8'hd3)^
({{8{data_in[195][6]}} & 8'h8d)^
({{8{data_in[195][7]}} & 8'h31)^
({{8{data_in[196][0]}} & 8'ha4)^
({{8{data_in[196][1]}} & 8'h63)^
({{8{data_in[196][2]}} & 8'hc6)^
({{8{data_in[196][3]}} & 8'ha7)^
({{8{data_in[196][4]}} & 8'h65)^
({{8{data_in[196][5]}} & 8'hca)^
({{8{data_in[196][6]}} & 8'hbf)^
({{8{data_in[196][7]}} & 8'h55)^
({{8{data_in[197][0]}} & 8'hb1)^
({{8{data_in[197][1]}} & 8'h49)^
({{8{data_in[197][2]}} & 8'h92)^
({{8{data_in[197][3]}} & 8'hf)^
({{8{data_in[197][4]}} & 8'h1e)^
({{8{data_in[197][5]}} & 8'h3c)^
({{8{data_in[197][6]}} & 8'h78)^
({{8{data_in[197][7]}} & 8'hf0)^
({{8{data_in[198][0]}} & 8'hfb)^
({{8{data_in[198][1]}} & 8'hdd)^
({{8{data_in[198][2]}} & 8'h91)^
({{8{data_in[198][3]}} & 8'h9)^
({{8{data_in[198][4]}} & 8'h12)^
({{8{data_in[198][5]}} & 8'h24)^
({{8{data_in[198][6]}} & 8'h48)^
({{8{data_in[198][7]}} & 8'h90)^
({{8{data_in[199][0]}} & 8'h5e)^
({{8{data_in[199][1]}} & 8'hbc)^
({{8{data_in[199][2]}} & 8'h53)^
({{8{data_in[199][3]}} & 8'ha6)^
({{8{data_in[199][4]}} & 8'h67)^
({{8{data_in[199][5]}} & 8'hce)^
({{8{data_in[199][6]}} & 8'hb7)^
({{8{data_in[199][7]}} & 8'h45)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[200][0]}} & 8'h1a)^
({{8{data_in[200][1]}} & 8'h34)^
({{8{data_in[200][2]}} & 8'h68)^
({{8{data_in[200][3]}} & 8'hd0)^
({{8{data_in[200][4]}} & 8'h8b)^
({{8{data_in[200][5]}} & 8'h3d)^
({{8{data_in[200][6]}} & 8'h7a)^
({{8{data_in[200][7]}} & 8'hf4)^
({{8{data_in[201][0]}} & 8'h3)^
({{8{data_in[201][1]}} & 8'h6)^
({{8{data_in[201][2]}} & 8'hc)^
({{8{data_in[201][3]}} & 8'h18)^
({{8{data_in[201][4]}} & 8'h30)^
({{8{data_in[201][5]}} & 8'h60)^
({{8{data_in[201][6]}} & 8'hc0)^
({{8{data_in[201][7]}} & 8'hab)^
({{8{data_in[202][0]}} & 8'h9a)^
({{8{data_in[202][1]}} & 8'h1f)^
({{8{data_in[202][2]}} & 8'h3e)^
({{8{data_in[202][3]}} & 8'h7c)^
({{8{data_in[202][4]}} & 8'hf8)^
({{8{data_in[202][5]}} & 8'hdb)^
({{8{data_in[202][6]}} & 8'h9d)^
({{8{data_in[202][7]}} & 8'h11)^
({{8{data_in[203][0]}} & 8'hb6)^
({{8{data_in[203][1]}} & 8'h47)^
({{8{data_in[203][2]}} & 8'h8e)^
({{8{data_in[203][3]}} & 8'h37)^
({{8{data_in[203][4]}} & 8'h6e)^
({{8{data_in[203][5]}} & 8'hdc)^
({{8{data_in[203][6]}} & 8'h93)^
({{8{data_in[203][7]}} & 8'hd)^
({{8{data_in[204][0]}} & 8'h49)^
({{8{data_in[204][1]}} & 8'h92)^
({{8{data_in[204][2]}} & 8'hf)^
({{8{data_in[204][3]}} & 8'h1e)^
({{8{data_in[204][4]}} & 8'h3c)^
({{8{data_in[204][5]}} & 8'h78)^
({{8{data_in[204][6]}} & 8'hf0)^
({{8{data_in[204][7]}} & 8'hcb)^
({{8{data_in[205][0]}} & 8'h33)^
({{8{data_in[205][1]}} & 8'h66)^
({{8{data_in[205][2]}} & 8'hcc)^
({{8{data_in[205][3]}} & 8'hb3)^
({{8{data_in[205][4]}} & 8'h4d)^
({{8{data_in[205][5]}} & 8'h9a)^
({{8{data_in[205][6]}} & 8'h1f)^
({{8{data_in[205][7]}} & 8'h3e)^
({{8{data_in[206][0]}} & 8'ha6)^
({{8{data_in[206][1]}} & 8'h67)^
({{8{data_in[206][2]}} & 8'hce)^
({{8{data_in[206][3]}} & 8'hb7)^
({{8{data_in[206][4]}} & 8'h45)^
({{8{data_in[206][5]}} & 8'h8a)^
({{8{data_in[206][6]}} & 8'h3f)^
({{8{data_in[206][7]}} & 8'h7e)^
({{8{data_in[207][0]}} & 8'haf)^
({{8{data_in[207][1]}} & 8'h75)^
({{8{data_in[207][2]}} & 8'hea)^
({{8{data_in[207][3]}} & 8'hff)^
({{8{data_in[207][4]}} & 8'hd5)^
({{8{data_in[207][5]}} & 8'h81)^
({{8{data_in[207][6]}} & 8'h29)^
({{8{data_in[207][7]}} & 8'h52)^
({{8{data_in[208][0]}} & 8'h23)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[208][1]}} & 8'h46)^
({{8{data_in[208][2]}} & 8'h8c)^
({{8{data_in[208][3]}} & 8'h33)^
({{8{data_in[208][4]}} & 8'h66)^
({{8{data_in[208][5]}} & 8'hcc)^
({{8{data_in[208][6]}} & 8'hb3)^
({{8{data_in[208][7]}} & 8'h4d)^
({{8{data_in[209][0]}} & 8'h67)^
({{8{data_in[209][1]}} & 8'hce)^
({{8{data_in[209][2]}} & 8'hb7)^
({{8{data_in[209][3]}} & 8'h45)^
({{8{data_in[209][4]}} & 8'h8a)^
({{8{data_in[209][5]}} & 8'h3f)^
({{8{data_in[209][6]}} & 8'h7e)^
({{8{data_in[209][7]}} & 8'hfc)^
({{8{data_in[210][0]}} & 8'h74)^
({{8{data_in[210][1]}} & 8'he8)^
({{8{data_in[210][2]}} & 8'hfb)^
({{8{data_in[210][3]}} & 8'hdd)^
({{8{data_in[210][4]}} & 8'h91)^
({{8{data_in[210][5]}} & 8'h9)^
({{8{data_in[210][6]}} & 8'h12)^
({{8{data_in[210][7]}} & 8'h24)^
({{8{data_in[211][0]}} & 8'h8f)^
({{8{data_in[211][1]}} & 8'h35)^
({{8{data_in[211][2]}} & 8'h6a)^
({{8{data_in[211][3]}} & 8'hd4)^
({{8{data_in[211][4]}} & 8'h83)^
({{8{data_in[211][5]}} & 8'h2d)^
({{8{data_in[211][6]}} & 8'h5a)^
({{8{data_in[211][7]}} & 8'hb4)^
({{8{data_in[212][0]}} & 8'h65)^
({{8{data_in[212][1]}} & 8'hca)^
({{8{data_in[212][2]}} & 8'hbf)^
({{8{data_in[212][3]}} & 8'h55)^
({{8{data_in[212][4]}} & 8'haa)^
({{8{data_in[212][5]}} & 8'h7f)^
({{8{data_in[212][6]}} & 8'hfe)^
({{8{data_in[212][7]}} & 8'hd7)^
({{8{data_in[213][0]}} & 8'h57)^
({{8{data_in[213][1]}} & 8'hae)^
({{8{data_in[213][2]}} & 8'h77)^
({{8{data_in[213][3]}} & 8'hee)^
({{8{data_in[213][4]}} & 8'hf7)^
({{8{data_in[213][5]}} & 8'hc5)^
({{8{data_in[213][6]}} & 8'ha1)^
({{8{data_in[213][7]}} & 8'h69)^
({{8{data_in[214][0]}} & 8'hf2)^
({{8{data_in[214][1]}} & 8'hcf)^
({{8{data_in[214][2]}} & 8'hb5)^
({{8{data_in[214][3]}} & 8'h41)^
({{8{data_in[214][4]}} & 8'h82)^
({{8{data_in[214][5]}} & 8'h2f)^
({{8{data_in[214][6]}} & 8'h5e)^
({{8{data_in[214][7]}} & 8'hbc)^
({{8{data_in[215][0]}} & 8'h9a)^
({{8{data_in[215][1]}} & 8'h1f)^
({{8{data_in[215][2]}} & 8'h3e)^
({{8{data_in[215][3]}} & 8'h7c)^
({{8{data_in[215][4]}} & 8'hf8)^
({{8{data_in[215][5]}} & 8'hdb)^
({{8{data_in[215][6]}} & 8'h9d)^
({{8{data_in[215][7]}} & 8'h11)^
({{8{data_in[216][0]}} & 8'h1a)^
({{8{data_in[216][1]}} & 8'h34)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[216][2]}} & 8'h68)^
({{8{data_in[216][3]}} & 8'hd0)^
({{8{data_in[216][4]}} & 8'h8b)^
({{8{data_in[216][5]}} & 8'h3d)^
({{8{data_in[216][6]}} & 8'h7a)^
({{8{data_in[216][7]}} & 8'hf4)^
({{8{data_in[217][0]}} & 8'h76)^
({{8{data_in[217][1]}} & 8'hec)^
({{8{data_in[217][2]}} & 8'hf3)^
({{8{data_in[217][3]}} & 8'hcd)^
({{8{data_in[217][4]}} & 8'hb1)^
({{8{data_in[217][5]}} & 8'h49)^
({{8{data_in[217][6]}} & 8'h92)^
({{8{data_in[217][7]}} & 8'hf)^
({{8{data_in[218][0]}} & 8'h46)^
({{8{data_in[218][1]}} & 8'h8c)^
({{8{data_in[218][2]}} & 8'h33)^
({{8{data_in[218][3]}} & 8'h66)^
({{8{data_in[218][4]}} & 8'hcc)^
({{8{data_in[218][5]}} & 8'hb3)^
({{8{data_in[218][6]}} & 8'h4d)^
({{8{data_in[218][7]}} & 8'h9a)^
({{8{data_in[219][0]}} & 8'h7a)^
({{8{data_in[219][1]}} & 8'hf4)^
({{8{data_in[219][2]}} & 8'hc3)^
({{8{data_in[219][3]}} & 8'had)^
({{8{data_in[219][4]}} & 8'h71)^
({{8{data_in[219][5]}} & 8'he2)^
({{8{data_in[219][6]}} & 8'hef)^
({{8{data_in[219][7]}} & 8'hf5)^
({{8{data_in[220][0]}} & 8'h6)^
({{8{data_in[220][1]}} & 8'hc)^
({{8{data_in[220][2]}} & 8'h18)^
({{8{data_in[220][3]}} & 8'h30)^
({{8{data_in[220][4]}} & 8'h60)^
({{8{data_in[220][5]}} & 8'hc0)^
({{8{data_in[220][6]}} & 8'hab)^
({{8{data_in[220][7]}} & 8'h7d)^
({{8{data_in[221][0]}} & 8'hc9)^
({{8{data_in[221][1]}} & 8'hb9)^
({{8{data_in[221][2]}} & 8'h59)^
({{8{data_in[221][3]}} & 8'hb2)^
({{8{data_in[221][4]}} & 8'h4f)^
({{8{data_in[221][5]}} & 8'h9e)^
({{8{data_in[221][6]}} & 8'h17)^
({{8{data_in[221][7]}} & 8'h2e)^
({{8{data_in[222][0]}} & 8'h50)^
({{8{data_in[222][1]}} & 8'ha0)^
({{8{data_in[222][2]}} & 8'h6b)^
({{8{data_in[222][3]}} & 8'hd6)^
({{8{data_in[222][4]}} & 8'h87)^
({{8{data_in[222][5]}} & 8'h25)^
({{8{data_in[222][6]}} & 8'h4a)^
({{8{data_in[222][7]}} & 8'h94)^
({{8{data_in[223][0]}} & 8'h38)^
({{8{data_in[223][1]}} & 8'h70)^
({{8{data_in[223][2]}} & 8'he0)^
({{8{data_in[223][3]}} & 8'heb)^
({{8{data_in[223][4]}} & 8'hfd)^
({{8{data_in[223][5]}} & 8'hd1)^
({{8{data_in[223][6]}} & 8'h89)^
({{8{data_in[223][7]}} & 8'h39)^
({{8{data_in[224][0]}} & 8'hae)^
({{8{data_in[224][1]}} & 8'h77)^
({{8{data_in[224][2]}} & 8'hee)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[224][3]}} & 8'hf7)^
({{8{data_in[224][4]}} & 8'hc5)^
({{8{data_in[224][5]}} & 8'ha1)^
({{8{data_in[224][6]}} & 8'h69)^
({{8{data_in[224][7]}} & 8'hd2)^
({{8{data_in[225][0]}} & 8'h34)^
({{8{data_in[225][1]}} & 8'h68)^
({{8{data_in[225][2]}} & 8'hd0)^
({{8{data_in[225][3]}} & 8'h8b)^
({{8{data_in[225][4]}} & 8'h3d)^
({{8{data_in[225][5]}} & 8'h7a)^
({{8{data_in[225][6]}} & 8'hf4)^
({{8{data_in[225][7]}} & 8'hc3)^
({{8{data_in[226][0]}} & 8'h9e)^
({{8{data_in[226][1]}} & 8'h17)^
({{8{data_in[226][2]}} & 8'h2e)^
({{8{data_in[226][3]}} & 8'h5c)^
({{8{data_in[226][4]}} & 8'hb8)^
({{8{data_in[226][5]}} & 8'h5b)^
({{8{data_in[226][6]}} & 8'hb6)^
({{8{data_in[226][7]}} & 8'h47)^
({{8{data_in[227][0]}} & 8'h5d)^
({{8{data_in[227][1]}} & 8'hba)^
({{8{data_in[227][2]}} & 8'h5f)^
({{8{data_in[227][3]}} & 8'hbe)^
({{8{data_in[227][4]}} & 8'h57)^
({{8{data_in[227][5]}} & 8'hae)^
({{8{data_in[227][6]}} & 8'h77)^
({{8{data_in[227][7]}} & 8'hee)^
({{8{data_in[228][0]}} & 8'hfe)^
({{8{data_in[228][1]}} & 8'hd7)^
({{8{data_in[228][2]}} & 8'h85)^
({{8{data_in[228][3]}} & 8'h21)^
({{8{data_in[228][4]}} & 8'h42)^
({{8{data_in[228][5]}} & 8'h84)^
({{8{data_in[228][6]}} & 8'h23)^
({{8{data_in[228][7]}} & 8'h46)^
({{8{data_in[229][0]}} & 8'hdc)^
({{8{data_in[229][1]}} & 8'h93)^
({{8{data_in[229][2]}} & 8'hd)^
({{8{data_in[229][3]}} & 8'h1a)^
({{8{data_in[229][4]}} & 8'h34)^
({{8{data_in[229][5]}} & 8'h68)^
({{8{data_in[229][6]}} & 8'hd0)^
({{8{data_in[229][7]}} & 8'h8b)^
({{8{data_in[230][0]}} & 8'hd9)^
({{8{data_in[230][1]}} & 8'h99)^
({{8{data_in[230][2]}} & 8'h19)^
({{8{data_in[230][3]}} & 8'h32)^
({{8{data_in[230][4]}} & 8'h64)^
({{8{data_in[230][5]}} & 8'hc8)^
({{8{data_in[230][6]}} & 8'hbb)^
({{8{data_in[230][7]}} & 8'h5d)^
({{8{data_in[231][0]}} & 8'hee)^
({{8{data_in[231][1]}} & 8'hf7)^
({{8{data_in[231][2]}} & 8'hc5)^
({{8{data_in[231][3]}} & 8'ha1)^
({{8{data_in[231][4]}} & 8'h69)^
({{8{data_in[231][5]}} & 8'hd2)^
({{8{data_in[231][6]}} & 8'h8f)^
({{8{data_in[231][7]}} & 8'h35)^
({{8{data_in[232][0]}} & 8'h39)^
({{8{data_in[232][1]}} & 8'h72)^
({{8{data_in[232][2]}} & 8'he4)^
({{8{data_in[232][3]}} & 8'he3)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[232][4]}} & 8'hd})^
({{8{data_in[232][5]}} & 8'hf1})^
({{8{data_in[232][6]}} & 8'hc9})^
({{8{data_in[232][7]}} & 8'hb9})^
({{8{data_in[233][0]}} & 8'h30})^
({{8{data_in[233][1]}} & 8'h60})^
({{8{data_in[233][2]}} & 8'hc0})^
({{8{data_in[233][3]}} & 8'hab})^
({{8{data_in[233][4]}} & 8'h7d})^
({{8{data_in[233][5]}} & 8'hfa})^
({{8{data_in[233][6]}} & 8'hdf})^
({{8{data_in[233][7]}} & 8'h95})^
({{8{data_in[234][0]}} & 8'hfd})^
({{8{data_in[234][1]}} & 8'hd1})^
({{8{data_in[234][2]}} & 8'h89})^
({{8{data_in[234][3]}} & 8'h39})^
({{8{data_in[234][4]}} & 8'h72})^
({{8{data_in[234][5]}} & 8'he4})^
({{8{data_in[234][6]}} & 8'he3})^
({{8{data_in[234][7]}} & 8'hd})^
({{8{data_in[235][0]}} & 8'h59})^
({{8{data_in[235][1]}} & 8'hb2})^
({{8{data_in[235][2]}} & 8'h4f})^
({{8{data_in[235][3]}} & 8'h9e})^
({{8{data_in[235][4]}} & 8'h17})^
({{8{data_in[235][5]}} & 8'h2e})^
({{8{data_in[235][6]}} & 8'h5c})^
({{8{data_in[235][7]}} & 8'hb8})^
({{8{data_in[236][0]}} & 8'hb6})^
({{8{data_in[236][1]}} & 8'h47})^
({{8{data_in[236][2]}} & 8'h8e})^
({{8{data_in[236][3]}} & 8'h37})^
({{8{data_in[236][4]}} & 8'h6e})^
({{8{data_in[236][5]}} & 8'hdc})^
({{8{data_in[236][6]}} & 8'h93})^
({{8{data_in[236][7]}} & 8'hd})^
({{8{data_in[237][0]}} & 8'hf4})^
({{8{data_in[237][1]}} & 8'hc3})^
({{8{data_in[237][2]}} & 8'had})^
({{8{data_in[237][3]}} & 8'h71})^
({{8{data_in[237][4]}} & 8'he2})^
({{8{data_in[237][5]}} & 8'hef})^
({{8{data_in[237][6]}} & 8'hf5})^
({{8{data_in[237][7]}} & 8'hc1})^
({{8{data_in[238][0]}} & 8'haa})^
({{8{data_in[238][1]}} & 8'h7f})^
({{8{data_in[238][2]}} & 8'hfe})^
({{8{data_in[238][3]}} & 8'hd7})^
({{8{data_in[238][4]}} & 8'h85})^
({{8{data_in[238][5]}} & 8'h21})^
({{8{data_in[238][6]}} & 8'h42})^
({{8{data_in[238][7]}} & 8'h84})^
({{8{data_in[239][0]}} & 8'h3f})^
({{8{data_in[239][1]}} & 8'h7e})^
({{8{data_in[239][2]}} & 8'hfc})^
({{8{data_in[239][3]}} & 8'hd3})^
({{8{data_in[239][4]}} & 8'h8d})^
({{8{data_in[239][5]}} & 8'h31})^
({{8{data_in[239][6]}} & 8'h62})^
({{8{data_in[239][7]}} & 8'hc4})^
({{8{data_in[240][0]}} & 8'hd2})^
({{8{data_in[240][1]}} & 8'h8f})^
({{8{data_in[240][2]}} & 8'h35})^
({{8{data_in[240][3]}} & 8'h6a})^
({{8{data_in[240][4]}} & 8'hd4})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[240][5]}} & 8'h83)^
({{8{data_in[240][6]}} & 8'h2d)^
({{8{data_in[240][7]}} & 8'h5a)^
({{8{data_in[241][0]}} & 8'h33)^
({{8{data_in[241][1]}} & 8'h66)^
({{8{data_in[241][2]}} & 8'hcc)^
({{8{data_in[241][3]}} & 8'hb3)^
({{8{data_in[241][4]}} & 8'h4d)^
({{8{data_in[241][5]}} & 8'h9a)^
({{8{data_in[241][6]}} & 8'h1f)^
({{8{data_in[241][7]}} & 8'h3e);

data_out[244] = ({8{data_in[0][0]}} & 8'hc3)^
({8{data_in[0][1]}} & 8'had)^
({8{data_in[0][2]}} & 8'h71)^
({8{data_in[0][3]}} & 8'he2)^
({8{data_in[0][4]}} & 8'hef)^
({8{data_in[0][5]}} & 8'hf5)^
({8{data_in[0][6]}} & 8'hc1)^
({8{data_in[0][7]}} & 8'ha9)^
({8{data_in[1][0]}} & 8'h1f)^
({8{data_in[1][1]}} & 8'h3e)^
({8{data_in[1][2]}} & 8'h7c)^
({8{data_in[1][3]}} & 8'hf8)^
({8{data_in[1][4]}} & 8'hdb)^
({8{data_in[1][5]}} & 8'h9d)^
({8{data_in[1][6]}} & 8'h11)^
({8{data_in[1][7]}} & 8'h22)^
({8{data_in[2][0]}} & 8'hac)^
({8{data_in[2][1]}} & 8'h73)^
({8{data_in[2][2]}} & 8'he6)^
({8{data_in[2][3]}} & 8'he7)^
({8{data_in[2][4]}} & 8'he5)^
({8{data_in[2][5]}} & 8'he1)^
({8{data_in[2][6]}} & 8'he9)^
({8{data_in[2][7]}} & 8'hf9)^
({8{data_in[3][0]}} & 8'h8c)^
({8{data_in[3][1]}} & 8'h33)^
({8{data_in[3][2]}} & 8'h66)^
({8{data_in[3][3]}} & 8'hcc)^
({8{data_in[3][4]}} & 8'hb3)^
({8{data_in[3][5]}} & 8'h4d)^
({8{data_in[3][6]}} & 8'h9a)^
({8{data_in[3][7]}} & 8'h1f)^
({8{data_in[4][0]}} & 8'h55)^
({8{data_in[4][1]}} & 8'haa)^
({8{data_in[4][2]}} & 8'h7f)^
({8{data_in[4][3]}} & 8'ffe)^
({8{data_in[4][4]}} & 8'hd7)^
({8{data_in[4][5]}} & 8'h85)^
({8{data_in[4][6]}} & 8'h21)^
({8{data_in[4][7]}} & 8'h42)^
({8{data_in[5][0]}} & 8'hdc)^
({8{data_in[5][1]}} & 8'h93)^
({8{data_in[5][2]}} & 8'hd)^
({8{data_in[5][3]}} & 8'h1a)^
({8{data_in[5][4]}} & 8'h34)^
({8{data_in[5][5]}} & 8'h68)^
({8{data_in[5][6]}} & 8'hd0)^
({8{data_in[5][7]}} & 8'h8b)^
({8{data_in[6][0]}} & 8'h3)^
({8{data_in[6][1]}} & 8'h6)^
({8{data_in[6][2]}} & 8'hc)^
({8{data_in[6][3]}} & 8'h18)^
({8{data_in[6][4]}} & 8'h30)^
({8{data_in[6][5]}} & 8'h60)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[6][6]}} & 8'hc0)^
({{8{data_in[6][7]}} & 8'hab)^
({{8{data_in[7][0]}} & 8'h1e)^
({{8{data_in[7][1]}} & 8'h3c)^
({{8{data_in[7][2]}} & 8'h78)^
({{8{data_in[7][3]}} & 8'hf0)^
({{8{data_in[7][4]}} & 8'hcb)^
({{8{data_in[7][5]}} & 8'hbd)^
({{8{data_in[7][6]}} & 8'h51)^
({{8{data_in[7][7]}} & 8'ha2)^
({{8{data_in[8][0]}} & 8'ha8)^
({{8{data_in[8][1]}} & 8'h7b)^
({{8{data_in[8][2]}} & 8'hf6)^
({{8{data_in[8][3]}} & 8'hc7)^
({{8{data_in[8][4]}} & 8'ha5)^
({{8{data_in[8][5]}} & 8'h61)^
({{8{data_in[8][6]}} & 8'hc2)^
({{8{data_in[8][7]}} & 8'haf)^
({{8{data_in[9][0]}} & 8'h17)^
({{8{data_in[9][1]}} & 8'h2e)^
({{8{data_in[9][2]}} & 8'h5c)^
({{8{data_in[9][3]}} & 8'hb8)^
({{8{data_in[9][4]}} & 8'h5b)^
({{8{data_in[9][5]}} & 8'hb6)^
({{8{data_in[9][6]}} & 8'h47)^
({{8{data_in[9][7]}} & 8'h8e)^
({{8{data_in[10][0]}} & 8'hab)^
({{8{data_in[10][1]}} & 8'h7d)^
({{8{data_in[10][2]}} & 8'hfa)^
({{8{data_in[10][3]}} & 8'hdf)^
({{8{data_in[10][4]}} & 8'h95)^
({{8{data_in[10][5]}} & 8'h1)^
({{8{data_in[10][6]}} & 8'h2)^
({{8{data_in[10][7]}} & 8'h4)^
({{8{data_in[11][0]}} & 8'h44)^
({{8{data_in[11][1]}} & 8'h88)^
({{8{data_in[11][2]}} & 8'h3b)^
({{8{data_in[11][3]}} & 8'h76)^
({{8{data_in[11][4]}} & 8'hec)^
({{8{data_in[11][5]}} & 8'hf3)^
({{8{data_in[11][6]}} & 8'hcd)^
({{8{data_in[11][7]}} & 8'hb1)^
({{8{data_in[12][0]}} & 8'hf6)^
({{8{data_in[12][1]}} & 8'hc7)^
({{8{data_in[12][2]}} & 8'ha5)^
({{8{data_in[12][3]}} & 8'h61)^
({{8{data_in[12][4]}} & 8'hc2)^
({{8{data_in[12][5]}} & 8'haf)^
({{8{data_in[12][6]}} & 8'h75)^
({{8{data_in[12][7]}} & 8'hea)^
({{8{data_in[13][0]}} & 8'hd5)^
({{8{data_in[13][1]}} & 8'h81)^
({{8{data_in[13][2]}} & 8'h29)^
({{8{data_in[13][3]}} & 8'h52)^
({{8{data_in[13][4]}} & 8'ha4)^
({{8{data_in[13][5]}} & 8'h63)^
({{8{data_in[13][6]}} & 8'hc6)^
({{8{data_in[13][7]}} & 8'ha7)^
({{8{data_in[14][0]}} & 8'h80)^
({{8{data_in[14][1]}} & 8'h2b)^
({{8{data_in[14][2]}} & 8'h56)^
({{8{data_in[14][3]}} & 8'hac)^
({{8{data_in[14][4]}} & 8'h73)^
({{8{data_in[14][5]}} & 8'he6)^
({{8{data_in[14][6]}} & 8'he7})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[14][7]}} & 8'he5)^
({{8{data_in[15][0]}} & 8'h19)^
({{8{data_in[15][1]}} & 8'h32)^
({{8{data_in[15][2]}} & 8'h64)^
({{8{data_in[15][3]}} & 8'hc8)^
({{8{data_in[15][4]}} & 8'hbb)^
({{8{data_in[15][5]}} & 8'h5d)^
({{8{data_in[15][6]}} & 8'hba)^
({{8{data_in[15][7]}} & 8'h5f)^
({{8{data_in[16][0]}} & 8'h8d)^
({{8{data_in[16][1]}} & 8'h31)^
({{8{data_in[16][2]}} & 8'h62)^
({{8{data_in[16][3]}} & 8'hc4)^
({{8{data_in[16][4]}} & 8'ha3)^
({{8{data_in[16][5]}} & 8'h6d)^
({{8{data_in[16][6]}} & 8'hda)^
({{8{data_in[16][7]}} & 8'h9f)^
({{8{data_in[17][0]}} & 8'h4c)^
({{8{data_in[17][1]}} & 8'h98)^
({{8{data_in[17][2]}} & 8'h1b)^
({{8{data_in[17][3]}} & 8'h36)^
({{8{data_in[17][4]}} & 8'h6c)^
({{8{data_in[17][5]}} & 8'hd8)^
({{8{data_in[17][6]}} & 8'h9b)^
({{8{data_in[17][7]}} & 8'h1d)^
({{8{data_in[18][0]}} & 8'h4a)^
({{8{data_in[18][1]}} & 8'h94)^
({{8{data_in[18][2]}} & 8'h3)^
({{8{data_in[18][3]}} & 8'h6)^
({{8{data_in[18][4]}} & 8'hc)^
({{8{data_in[18][5]}} & 8'h18)^
({{8{data_in[18][6]}} & 8'h30)^
({{8{data_in[18][7]}} & 8'h60)^
({{8{data_in[19][0]}} & 8'h1e)^
({{8{data_in[19][1]}} & 8'h3c)^
({{8{data_in[19][2]}} & 8'h78)^
({{8{data_in[19][3]}} & 8'hf0)^
({{8{data_in[19][4]}} & 8'hcb)^
({{8{data_in[19][5]}} & 8'hbd)^
({{8{data_in[19][6]}} & 8'h51)^
({{8{data_in[19][7]}} & 8'ha2)^
({{8{data_in[20][0]}} & 8'ha2)^
({{8{data_in[20][1]}} & 8'h6f)^
({{8{data_in[20][2]}} & 8'hde)^
({{8{data_in[20][3]}} & 8'h97)^
({{8{data_in[20][4]}} & 8'h5)^
({{8{data_in[20][5]}} & 8'ha)^
({{8{data_in[20][6]}} & 8'h14)^
({{8{data_in[20][7]}} & 8'h28)^
({{8{data_in[21][0]}} & 8'h73)^
({{8{data_in[21][1]}} & 8'he6)^
({{8{data_in[21][2]}} & 8'he7)^
({{8{data_in[21][3]}} & 8'he5)^
({{8{data_in[21][4]}} & 8'he1)^
({{8{data_in[21][5]}} & 8'he9)^
({{8{data_in[21][6]}} & 8'hf9)^
({{8{data_in[21][7]}} & 8'hd9)^
({{8{data_in[22][0]}} & 8'hb3)^
({{8{data_in[22][1]}} & 8'h4d)^
({{8{data_in[22][2]}} & 8'h9a)^
({{8{data_in[22][3]}} & 8'h1f)^
({{8{data_in[22][4]}} & 8'h3e)^
({{8{data_in[22][5]}} & 8'h7c)^
({{8{data_in[22][6]}} & 8'hf8)^
({{8{data_in[22][7]}} & 8'hdb)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[23][0]}} & 8'he)^
({{8{data_in[23][1]}} & 8'h1c)^
({{8{data_in[23][2]}} & 8'h38)^
({{8{data_in[23][3]}} & 8'h70)^
({{8{data_in[23][4]}} & 8'he0)^
({{8{data_in[23][5]}} & 8'heb)^
({{8{data_in[23][6]}} & 8'hfd)^
({{8{data_in[23][7]}} & 8'hd1)^
({{8{data_in[24][0]}} & 8'h2d)^
({{8{data_in[24][1]}} & 8'h5a)^
({{8{data_in[24][2]}} & 8'hb4)^
({{8{data_in[24][3]}} & 8'h43)^
({{8{data_in[24][4]}} & 8'h86)^
({{8{data_in[24][5]}} & 8'h27)^
({{8{data_in[24][6]}} & 8'h4e)^
({{8{data_in[24][7]}} & 8'h9c)^
({{8{data_in[25][0]}} & 8'h46)^
({{8{data_in[25][1]}} & 8'h8c)^
({{8{data_in[25][2]}} & 8'h33)^
({{8{data_in[25][3]}} & 8'h66)^
({{8{data_in[25][4]}} & 8'hcc)^
({{8{data_in[25][5]}} & 8'hb3)^
({{8{data_in[25][6]}} & 8'h4d)^
({{8{data_in[25][7]}} & 8'h9a)^
({{8{data_in[26][0]}} & 8'hb2)^
({{8{data_in[26][1]}} & 8'h4f)^
({{8{data_in[26][2]}} & 8'h9e)^
({{8{data_in[26][3]}} & 8'h17)^
({{8{data_in[26][4]}} & 8'h2e)^
({{8{data_in[26][5]}} & 8'h5c)^
({{8{data_in[26][6]}} & 8'hb8)^
({{8{data_in[26][7]}} & 8'h5b)^
({{8{data_in[27][0]}} & 8'h8b)^
({{8{data_in[27][1]}} & 8'h3d)^
({{8{data_in[27][2]}} & 8'h7a)^
({{8{data_in[27][3]}} & 8'hf4)^
({{8{data_in[27][4]}} & 8'hc3)^
({{8{data_in[27][5]}} & 8'had)^
({{8{data_in[27][6]}} & 8'h71)^
({{8{data_in[27][7]}} & 8'he2)^
({{8{data_in[28][0]}} & 8'h87)^
({{8{data_in[28][1]}} & 8'h25)^
({{8{data_in[28][2]}} & 8'h4a)^
({{8{data_in[28][3]}} & 8'h94)^
({{8{data_in[28][4]}} & 8'h3)^
({{8{data_in[28][5]}} & 8'h6)^
({{8{data_in[28][6]}} & 8'hc)^
({{8{data_in[28][7]}} & 8'h18)^
({{8{data_in[29][0]}} & 8'h65)^
({{8{data_in[29][1]}} & 8'hca)^
({{8{data_in[29][2]}} & 8'hbf)^
({{8{data_in[29][3]}} & 8'h55)^
({{8{data_in[29][4]}} & 8'haa)^
({{8{data_in[29][5]}} & 8'h7f)^
({{8{data_in[29][6]}} & 8'hfe)^
({{8{data_in[29][7]}} & 8'hd7)^
({{8{data_in[30][0]}} & 8'h13)^
({{8{data_in[30][1]}} & 8'h26)^
({{8{data_in[30][2]}} & 8'h4c)^
({{8{data_in[30][3]}} & 8'h98)^
({{8{data_in[30][4]}} & 8'h1b)^
({{8{data_in[30][5]}} & 8'h36)^
({{8{data_in[30][6]}} & 8'h6c)^
({{8{data_in[30][7]}} & 8'hd8)^
({{8{data_in[31][0]}} & 8'h5d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[31][1]}} & 8'hba)^
({{8{data_in[31][2]}} & 8'h5f)^
({{8{data_in[31][3]}} & 8'hbe)^
({{8{data_in[31][4]}} & 8'h57)^
({{8{data_in[31][5]}} & 8'hae)^
({{8{data_in[31][6]}} & 8'h77)^
({{8{data_in[31][7]}} & 8'hee)^
({{8{data_in[32][0]}} & 8'h41)^
({{8{data_in[32][1]}} & 8'h82)^
({{8{data_in[32][2]}} & 8'h2f)^
({{8{data_in[32][3]}} & 8'h5e)^
({{8{data_in[32][4]}} & 8'hbc)^
({{8{data_in[32][5]}} & 8'h53)^
({{8{data_in[32][6]}} & 8'ha6)^
({{8{data_in[32][7]}} & 8'h67)^
({{8{data_in[33][0]}} & 8'hd9)^
({{8{data_in[33][1]}} & 8'h99)^
({{8{data_in[33][2]}} & 8'h19)^
({{8{data_in[33][3]}} & 8'h32)^
({{8{data_in[33][4]}} & 8'h64)^
({{8{data_in[33][5]}} & 8'hc8)^
({{8{data_in[33][6]}} & 8'hbb)^
({{8{data_in[33][7]}} & 8'h5d)^
({{8{data_in[34][0]}} & 8'hff)^
({{8{data_in[34][1]}} & 8'hd5)^
({{8{data_in[34][2]}} & 8'h81)^
({{8{data_in[34][3]}} & 8'h29)^
({{8{data_in[34][4]}} & 8'h52)^
({{8{data_in[34][5]}} & 8'ha4)^
({{8{data_in[34][6]}} & 8'h63)^
({{8{data_in[34][7]}} & 8'hc6)^
({{8{data_in[35][0]}} & 8'h35)^
({{8{data_in[35][1]}} & 8'h6a)^
({{8{data_in[35][2]}} & 8'hd4)^
({{8{data_in[35][3]}} & 8'h83)^
({{8{data_in[35][4]}} & 8'h2d)^
({{8{data_in[35][5]}} & 8'h5a)^
({{8{data_in[35][6]}} & 8'hb4)^
({{8{data_in[35][7]}} & 8'h43)^
({{8{data_in[36][0]}} & 8'hbf)^
({{8{data_in[36][1]}} & 8'h55)^
({{8{data_in[36][2]}} & 8'haa)^
({{8{data_in[36][3]}} & 8'h7f)^
({{8{data_in[36][4]}} & 8'hfe)^
({{8{data_in[36][5]}} & 8'hd7)^
({{8{data_in[36][6]}} & 8'h85)^
({{8{data_in[36][7]}} & 8'h21)^
({{8{data_in[37][0]}} & 8'h49)^
({{8{data_in[37][1]}} & 8'h92)^
({{8{data_in[37][2]}} & 8'hf)^
({{8{data_in[37][3]}} & 8'h1e)^
({{8{data_in[37][4]}} & 8'h3c)^
({{8{data_in[37][5]}} & 8'h78)^
({{8{data_in[37][6]}} & 8'hf0)^
({{8{data_in[37][7]}} & 8'hcb)^
({{8{data_in[38][0]}} & 8'h3f)^
({{8{data_in[38][1]}} & 8'h7e)^
({{8{data_in[38][2]}} & 8'hfc)^
({{8{data_in[38][3]}} & 8'hd3)^
({{8{data_in[38][4]}} & 8'h8d)^
({{8{data_in[38][5]}} & 8'h31)^
({{8{data_in[38][6]}} & 8'h62)^
({{8{data_in[38][7]}} & 8'hc4)^
({{8{data_in[39][0]}} & 8'hee)^
({{8{data_in[39][1]}} & 8'hf7})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[39][2]}} & 8'hc5)^
({{8{data_in[39][3]}} & 8'ha1)^
({{8{data_in[39][4]}} & 8'h69)^
({{8{data_in[39][5]}} & 8'hd2)^
({{8{data_in[39][6]}} & 8'h8f)^
({{8{data_in[39][7]}} & 8'h35)^
({{8{data_in[40][0]}} & 8'h64)^
({{8{data_in[40][1]}} & 8'hc8)^
({{8{data_in[40][2]}} & 8'hbb)^
({{8{data_in[40][3]}} & 8'h5d)^
({{8{data_in[40][4]}} & 8'hba)^
({{8{data_in[40][5]}} & 8'h5f)^
({{8{data_in[40][6]}} & 8'hbe)^
({{8{data_in[40][7]}} & 8'h57)^
({{8{data_in[41][0]}} & 8'h13)^
({{8{data_in[41][1]}} & 8'h26)^
({{8{data_in[41][2]}} & 8'h4c)^
({{8{data_in[41][3]}} & 8'h98)^
({{8{data_in[41][4]}} & 8'h1b)^
({{8{data_in[41][5]}} & 8'h36)^
({{8{data_in[41][6]}} & 8'h6c)^
({{8{data_in[41][7]}} & 8'hd8)^
({{8{data_in[42][0]}} & 8'h91)^
({{8{data_in[42][1]}} & 8'h9)^
({{8{data_in[42][2]}} & 8'h12)^
({{8{data_in[42][3]}} & 8'h24)^
({{8{data_in[42][4]}} & 8'h48)^
({{8{data_in[42][5]}} & 8'h90)^
({{8{data_in[42][6]}} & 8'hb)^
({{8{data_in[42][7]}} & 8'h16)^
({{8{data_in[43][0]}} & 8'hdf)^
({{8{data_in[43][1]}} & 8'h95)^
({{8{data_in[43][2]}} & 8'h1)^
({{8{data_in[43][3]}} & 8'h2)^
({{8{data_in[43][4]}} & 8'h4)^
({{8{data_in[43][5]}} & 8'h8)^
({{8{data_in[43][6]}} & 8'h10)^
({{8{data_in[43][7]}} & 8'h20)^
({{8{data_in[44][0]}} & 8'h91)^
({{8{data_in[44][1]}} & 8'h9)^
({{8{data_in[44][2]}} & 8'h12)^
({{8{data_in[44][3]}} & 8'h24)^
({{8{data_in[44][4]}} & 8'h48)^
({{8{data_in[44][5]}} & 8'h90)^
({{8{data_in[44][6]}} & 8'hb)^
({{8{data_in[44][7]}} & 8'h16)^
({{8{data_in[45][0]}} & 8'hde)^
({{8{data_in[45][1]}} & 8'h97)^
({{8{data_in[45][2]}} & 8'h5)^
({{8{data_in[45][3]}} & 8'ha)^
({{8{data_in[45][4]}} & 8'h14)^
({{8{data_in[45][5]}} & 8'h28)^
({{8{data_in[45][6]}} & 8'h50)^
({{8{data_in[45][7]}} & 8'ha0)^
({{8{data_in[46][0]}} & 8'hc8)^
({{8{data_in[46][1]}} & 8'hbb)^
({{8{data_in[46][2]}} & 8'h5d)^
({{8{data_in[46][3]}} & 8'hba)^
({{8{data_in[46][4]}} & 8'h5f)^
({{8{data_in[46][5]}} & 8'hbe)^
({{8{data_in[46][6]}} & 8'h57)^
({{8{data_in[46][7]}} & 8'hae)^
({{8{data_in[47][0]}} & 8'h22)^
({{8{data_in[47][1]}} & 8'h44)^
({{8{data_in[47][2]}} & 8'h88)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[47][3]}} & 8'h3b)^
({{8{data_in[47][4]}} & 8'h76)^
({{8{data_in[47][5]}} & 8'hec)^
({{8{data_in[47][6]}} & 8'hf3)^
({{8{data_in[47][7]}} & 8'hcd)^
({{8{data_in[48][0]}} & 8'h49)^
({{8{data_in[48][1]}} & 8'h92)^
({{8{data_in[48][2]}} & 8'hf)^
({{8{data_in[48][3]}} & 8'h1e)^
({{8{data_in[48][4]}} & 8'h3c)^
({{8{data_in[48][5]}} & 8'h78)^
({{8{data_in[48][6]}} & 8'hf0)^
({{8{data_in[48][7]}} & 8'hcb)^
({{8{data_in[49][0]}} & 8'hd8)^
({{8{data_in[49][1]}} & 8'h9b)^
({{8{data_in[49][2]}} & 8'h1d)^
({{8{data_in[49][3]}} & 8'h3a)^
({{8{data_in[49][4]}} & 8'h74)^
({{8{data_in[49][5]}} & 8'he8)^
({{8{data_in[49][6]}} & 8'fib)^
({{8{data_in[49][7]}} & 8'hdd)^
({{8{data_in[50][0]}} & 8'h39)^
({{8{data_in[50][1]}} & 8'h72)^
({{8{data_in[50][2]}} & 8'he4)^
({{8{data_in[50][3]}} & 8'he3)^
({{8{data_in[50][4]}} & 8'hed)^
({{8{data_in[50][5]}} & 8'hf1)^
({{8{data_in[50][6]}} & 8'hc9)^
({{8{data_in[50][7]}} & 8'hb9)^
({{8{data_in[51][0]}} & 8'h83)^
({{8{data_in[51][1]}} & 8'h2d)^
({{8{data_in[51][2]}} & 8'h5a)^
({{8{data_in[51][3]}} & 8'hb4)^
({{8{data_in[51][4]}} & 8'h43)^
({{8{data_in[51][5]}} & 8'h86)^
({{8{data_in[51][6]}} & 8'h27)^
({{8{data_in[51][7]}} & 8'h4e)^
({{8{data_in[52][0]}} & 8'h37)^
({{8{data_in[52][1]}} & 8'h6e)^
({{8{data_in[52][2]}} & 8'hdc)^
({{8{data_in[52][3]}} & 8'h93)^
({{8{data_in[52][4]}} & 8'hd)^
({{8{data_in[52][5]}} & 8'h1a)^
({{8{data_in[52][6]}} & 8'h34)^
({{8{data_in[52][7]}} & 8'h68)^
({{8{data_in[53][0]}} & 8'hbb)^
({{8{data_in[53][1]}} & 8'h5d)^
({{8{data_in[53][2]}} & 8'hba)^
({{8{data_in[53][3]}} & 8'h5f)^
({{8{data_in[53][4]}} & 8'hbe)^
({{8{data_in[53][5]}} & 8'h57)^
({{8{data_in[53][6]}} & 8'hae)^
({{8{data_in[53][7]}} & 8'h77)^
({{8{data_in[54][0]}} & 8'h68)^
({{8{data_in[54][1]}} & 8'hd0)^
({{8{data_in[54][2]}} & 8'h8b)^
({{8{data_in[54][3]}} & 8'h3d)^
({{8{data_in[54][4]}} & 8'h7a)^
({{8{data_in[54][5]}} & 8'hf4)^
({{8{data_in[54][6]}} & 8'hc3)^
({{8{data_in[54][7]}} & 8'had)^
({{8{data_in[55][0]}} & 8'h5)^
({{8{data_in[55][1]}} & 8'ha)^
({{8{data_in[55][2]}} & 8'h14)^
({{8{data_in[55][3]}} & 8'h28)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[55][4]}} & 8'h50)^
({{8{data_in[55][5]}} & 8'ha0)^
({{8{data_in[55][6]}} & 8'h6b)^
({{8{data_in[55][7]}} & 8'hd6)^
({{8{data_in[56][0]}} & 8'he9)^
({{8{data_in[56][1]}} & 8'hf9)^
({{8{data_in[56][2]}} & 8'hd9)^
({{8{data_in[56][3]}} & 8'h99)^
({{8{data_in[56][4]}} & 8'h19)^
({{8{data_in[56][5]}} & 8'h32)^
({{8{data_in[56][6]}} & 8'h64)^
({{8{data_in[56][7]}} & 8'hc8)^
({{8{data_in[57][0]}} & 8'hcc)^
({{8{data_in[57][1]}} & 8'hb3)^
({{8{data_in[57][2]}} & 8'h4d)^
({{8{data_in[57][3]}} & 8'h9a)^
({{8{data_in[57][4]}} & 8'h1f)^
({{8{data_in[57][5]}} & 8'h3e)^
({{8{data_in[57][6]}} & 8'h7c)^
({{8{data_in[57][7]}} & 8'hf8)^
({{8{data_in[58][0]}} & 8'hbb)^
({{8{data_in[58][1]}} & 8'h5d)^
({{8{data_in[58][2]}} & 8'hba)^
({{8{data_in[58][3]}} & 8'h5f)^
({{8{data_in[58][4]}} & 8'hbe)^
({{8{data_in[58][5]}} & 8'h57)^
({{8{data_in[58][6]}} & 8'hae)^
({{8{data_in[58][7]}} & 8'h77)^
({{8{data_in[59][0]}} & 8'hfb)^
({{8{data_in[59][1]}} & 8'hdd)^
({{8{data_in[59][2]}} & 8'h91)^
({{8{data_in[59][3]}} & 8'h9)^
({{8{data_in[59][4]}} & 8'h12)^
({{8{data_in[59][5]}} & 8'h24)^
({{8{data_in[59][6]}} & 8'h48)^
({{8{data_in[59][7]}} & 8'h90)^
({{8{data_in[60][0]}} & 8'hdf)^
({{8{data_in[60][1]}} & 8'h95)^
({{8{data_in[60][2]}} & 8'h1)^
({{8{data_in[60][3]}} & 8'h2)^
({{8{data_in[60][4]}} & 8'h4)^
({{8{data_in[60][5]}} & 8'h8)^
({{8{data_in[60][6]}} & 8'h10)^
({{8{data_in[60][7]}} & 8'h20)^
({{8{data_in[61][0]}} & 8'hbe)^
({{8{data_in[61][1]}} & 8'h57)^
({{8{data_in[61][2]}} & 8'hae)^
({{8{data_in[61][3]}} & 8'h77)^
({{8{data_in[61][4]}} & 8'hee)^
({{8{data_in[61][5]}} & 8'hf7)^
({{8{data_in[61][6]}} & 8'hc5)^
({{8{data_in[61][7]}} & 8'ha1)^
({{8{data_in[62][0]}} & 8'h5e)^
({{8{data_in[62][1]}} & 8'hbc)^
({{8{data_in[62][2]}} & 8'h53)^
({{8{data_in[62][3]}} & 8'ha6)^
({{8{data_in[62][4]}} & 8'h67)^
({{8{data_in[62][5]}} & 8'hce)^
({{8{data_in[62][6]}} & 8'hb7)^
({{8{data_in[62][7]}} & 8'h45)^
({{8{data_in[63][0]}} & 8'h42)^
({{8{data_in[63][1]}} & 8'h84)^
({{8{data_in[63][2]}} & 8'h23)^
({{8{data_in[63][3]}} & 8'h46)^
({{8{data_in[63][4]}} & 8'h8c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[63][5]}} & 8'h33)^
({{8{data_in[63][6]}} & 8'h66)^
({{8{data_in[63][7]}} & 8'hcc)^
({{8{data_in[64][0]}} & 8'hf9)^
({{8{data_in[64][1]}} & 8'hd9)^
({{8{data_in[64][2]}} & 8'h99)^
({{8{data_in[64][3]}} & 8'h19)^
({{8{data_in[64][4]}} & 8'h32)^
({{8{data_in[64][5]}} & 8'h64)^
({{8{data_in[64][6]}} & 8'hc8)^
({{8{data_in[64][7]}} & 8'hbb)^
({{8{data_in[65][0]}} & 8'hc4)^
({{8{data_in[65][1]}} & 8'ha3)^
({{8{data_in[65][2]}} & 8'h6d)^
({{8{data_in[65][3]}} & 8'hda)^
({{8{data_in[65][4]}} & 8'h9f)^
({{8{data_in[65][5]}} & 8'h15)^
({{8{data_in[65][6]}} & 8'h2a)^
({{8{data_in[65][7]}} & 8'h54)^
({{8{data_in[66][0]}} & 8'hef)^
({{8{data_in[66][1]}} & 8'hf5)^
({{8{data_in[66][2]}} & 8'hc1)^
({{8{data_in[66][3]}} & 8'ha9)^
({{8{data_in[66][4]}} & 8'h79)^
({{8{data_in[66][5]}} & 8'hf2)^
({{8{data_in[66][6]}} & 8'hcf)^
({{8{data_in[66][7]}} & 8'hb5)^
({{8{data_in[67][0]}} & 8'h13)^
({{8{data_in[67][1]}} & 8'h26)^
({{8{data_in[67][2]}} & 8'h4c)^
({{8{data_in[67][3]}} & 8'h98)^
({{8{data_in[67][4]}} & 8'h1b)^
({{8{data_in[67][5]}} & 8'h36)^
({{8{data_in[67][6]}} & 8'h6c)^
({{8{data_in[67][7]}} & 8'hd8)^
({{8{data_in[68][0]}} & 8'hbc)^
({{8{data_in[68][1]}} & 8'h53)^
({{8{data_in[68][2]}} & 8'ha6)^
({{8{data_in[68][3]}} & 8'h67)^
({{8{data_in[68][4]}} & 8'hce)^
({{8{data_in[68][5]}} & 8'hb7)^
({{8{data_in[68][6]}} & 8'h45)^
({{8{data_in[68][7]}} & 8'h8a)^
({{8{data_in[69][0]}} & 8'h31)^
({{8{data_in[69][1]}} & 8'h62)^
({{8{data_in[69][2]}} & 8'hc4)^
({{8{data_in[69][3]}} & 8'ha3)^
({{8{data_in[69][4]}} & 8'h6d)^
({{8{data_in[69][5]}} & 8'hda)^
({{8{data_in[69][6]}} & 8'h9f)^
({{8{data_in[69][7]}} & 8'h15)^
({{8{data_in[70][0]}} & 8'hca)^
({{8{data_in[70][1]}} & 8'hbf)^
({{8{data_in[70][2]}} & 8'h55)^
({{8{data_in[70][3]}} & 8'haa)^
({{8{data_in[70][4]}} & 8'h7f)^
({{8{data_in[70][5]}} & 8'hfe)^
({{8{data_in[70][6]}} & 8'hd7)^
({{8{data_in[70][7]}} & 8'h85)^
({{8{data_in[71][0]}} & 8'hec)^
({{8{data_in[71][1]}} & 8'hf3)^
({{8{data_in[71][2]}} & 8'hcd)^
({{8{data_in[71][3]}} & 8'hb1)^
({{8{data_in[71][4]}} & 8'h49)^
({{8{data_in[71][5]}} & 8'h92})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[71][6]}} & 8'hf)^
({{8{data_in[71][7]}} & 8'h1e)^
({{8{data_in[72][0]}} & 8'h2a)^
({{8{data_in[72][1]}} & 8'h54)^
({{8{data_in[72][2]}} & 8'ha8)^
({{8{data_in[72][3]}} & 8'h7b)^
({{8{data_in[72][4]}} & 8'hf6)^
({{8{data_in[72][5]}} & 8'hc7)^
({{8{data_in[72][6]}} & 8'ha5)^
({{8{data_in[72][7]}} & 8'h61)^
({{8{data_in[73][0]}} & 8'h6f)^
({{8{data_in[73][1]}} & 8'hde)^
({{8{data_in[73][2]}} & 8'h97)^
({{8{data_in[73][3]}} & 8'h5)^
({{8{data_in[73][4]}} & 8'ha)^
({{8{data_in[73][5]}} & 8'h14)^
({{8{data_in[73][6]}} & 8'h28)^
({{8{data_in[73][7]}} & 8'h50)^
({{8{data_in[74][0]}} & 8'hfb)^
({{8{data_in[74][1]}} & 8'hdd)^
({{8{data_in[74][2]}} & 8'h91)^
({{8{data_in[74][3]}} & 8'h9)^
({{8{data_in[74][4]}} & 8'h12)^
({{8{data_in[74][5]}} & 8'h24)^
({{8{data_in[74][6]}} & 8'h48)^
({{8{data_in[74][7]}} & 8'h90)^
({{8{data_in[75][0]}} & 8'he6)^
({{8{data_in[75][1]}} & 8'he7)^
({{8{data_in[75][2]}} & 8'he5)^
({{8{data_in[75][3]}} & 8'he1)^
({{8{data_in[75][4]}} & 8'he9)^
({{8{data_in[75][5]}} & 8'hf9)^
({{8{data_in[75][6]}} & 8'hd9)^
({{8{data_in[75][7]}} & 8'h99)^
({{8{data_in[76][0]}} & 8'h82)^
({{8{data_in[76][1]}} & 8'h2f)^
({{8{data_in[76][2]}} & 8'h5e)^
({{8{data_in[76][3]}} & 8'hbc)^
({{8{data_in[76][4]}} & 8'h53)^
({{8{data_in[76][5]}} & 8'ha6)^
({{8{data_in[76][6]}} & 8'h67)^
({{8{data_in[76][7]}} & 8'hce)^
({{8{data_in[77][0]}} & 8'hd3)^
({{8{data_in[77][1]}} & 8'h8d)^
({{8{data_in[77][2]}} & 8'h31)^
({{8{data_in[77][3]}} & 8'h62)^
({{8{data_in[77][4]}} & 8'hc4)^
({{8{data_in[77][5]}} & 8'ha3)^
({{8{data_in[77][6]}} & 8'h6d)^
({{8{data_in[77][7]}} & 8'hda)^
({{8{data_in[78][0]}} & 8'h4c)^
({{8{data_in[78][1]}} & 8'h98)^
({{8{data_in[78][2]}} & 8'h1b)^
({{8{data_in[78][3]}} & 8'h36)^
({{8{data_in[78][4]}} & 8'h6c)^
({{8{data_in[78][5]}} & 8'hd8)^
({{8{data_in[78][6]}} & 8'h9b)^
({{8{data_in[78][7]}} & 8'h1d)^
({{8{data_in[79][0]}} & 8'hfe)^
({{8{data_in[79][1]}} & 8'hd7)^
({{8{data_in[79][2]}} & 8'h85)^
({{8{data_in[79][3]}} & 8'h21)^
({{8{data_in[79][4]}} & 8'h42)^
({{8{data_in[79][5]}} & 8'h84)^
({{8{data_in[79][6]}} & 8'h23})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[79][7]}} & 8'h46)^
({{8{data_in[80][0]}} & 8'hc1)^
({{8{data_in[80][1]}} & 8'ha9)^
({{8{data_in[80][2]}} & 8'h79)^
({{8{data_in[80][3]}} & 8'hf2)^
({{8{data_in[80][4]}} & 8'hcf)^
({{8{data_in[80][5]}} & 8'hb5)^
({{8{data_in[80][6]}} & 8'h41)^
({{8{data_in[80][7]}} & 8'h82)^
({{8{data_in[81][0]}} & 8'h12)^
({{8{data_in[81][1]}} & 8'h24)^
({{8{data_in[81][2]}} & 8'h48)^
({{8{data_in[81][3]}} & 8'h90)^
({{8{data_in[81][4]}} & 8'hb)^
({{8{data_in[81][5]}} & 8'h16)^
({{8{data_in[81][6]}} & 8'h2c)^
({{8{data_in[81][7]}} & 8'h58)^
({{8{data_in[82][0]}} & 8'hd2)^
({{8{data_in[82][1]}} & 8'h8f)^
({{8{data_in[82][2]}} & 8'h35)^
({{8{data_in[82][3]}} & 8'h6a)^
({{8{data_in[82][4]}} & 8'hd4)^
({{8{data_in[82][5]}} & 8'h83)^
({{8{data_in[82][6]}} & 8'h2d)^
({{8{data_in[82][7]}} & 8'h5a)^
({{8{data_in[83][0]}} & 8'h52)^
({{8{data_in[83][1]}} & 8'ha4)^
({{8{data_in[83][2]}} & 8'h63)^
({{8{data_in[83][3]}} & 8'hc6)^
({{8{data_in[83][4]}} & 8'ha7)^
({{8{data_in[83][5]}} & 8'h65)^
({{8{data_in[83][6]}} & 8'hca)^
({{8{data_in[83][7]}} & 8'hbf)^
({{8{data_in[84][0]}} & 8'h4e)^
({{8{data_in[84][1]}} & 8'h9c)^
({{8{data_in[84][2]}} & 8'h13)^
({{8{data_in[84][3]}} & 8'h26)^
({{8{data_in[84][4]}} & 8'h4c)^
({{8{data_in[84][5]}} & 8'h98)^
({{8{data_in[84][6]}} & 8'h1b)^
({{8{data_in[84][7]}} & 8'h36)^
({{8{data_in[85][0]}} & 8'ha6)^
({{8{data_in[85][1]}} & 8'h67)^
({{8{data_in[85][2]}} & 8'hce)^
({{8{data_in[85][3]}} & 8'hb7)^
({{8{data_in[85][4]}} & 8'h45)^
({{8{data_in[85][5]}} & 8'h8a)^
({{8{data_in[85][6]}} & 8'h3f)^
({{8{data_in[85][7]}} & 8'h7e)^
({{8{data_in[86][0]}} & 8'h9f)^
({{8{data_in[86][1]}} & 8'h15)^
({{8{data_in[86][2]}} & 8'h2a)^
({{8{data_in[86][3]}} & 8'h54)^
({{8{data_in[86][4]}} & 8'ha8)^
({{8{data_in[86][5]}} & 8'h7b)^
({{8{data_in[86][6]}} & 8'hf6)^
({{8{data_in[86][7]}} & 8'hc7)^
({{8{data_in[87][0]}} & 8'hef)^
({{8{data_in[87][1]}} & 8'hf5)^
({{8{data_in[87][2]}} & 8'hc1)^
({{8{data_in[87][3]}} & 8'ha9)^
({{8{data_in[87][4]}} & 8'h79)^
({{8{data_in[87][5]}} & 8'hf2)^
({{8{data_in[87][6]}} & 8'hcf)^
({{8{data_in[87][7]}} & 8'hb5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[88][0]}} & 8'h62)^
({{8{data_in[88][1]}} & 8'hc4)^
({{8{data_in[88][2]}} & 8'ha3)^
({{8{data_in[88][3]}} & 8'h6d)^
({{8{data_in[88][4]}} & 8'hda)^
({{8{data_in[88][5]}} & 8'h9f)^
({{8{data_in[88][6]}} & 8'h15)^
({{8{data_in[88][7]}} & 8'h2a)^
({{8{data_in[89][0]}} & 8'hfd)^
({{8{data_in[89][1]}} & 8'hd1)^
({{8{data_in[89][2]}} & 8'h89)^
({{8{data_in[89][3]}} & 8'h39)^
({{8{data_in[89][4]}} & 8'h72)^
({{8{data_in[89][5]}} & 8'he4)^
({{8{data_in[89][6]}} & 8'he3)^
({{8{data_in[89][7]}} & 8'hed)^
({{8{data_in[90][0]}} & 8'h76)^
({{8{data_in[90][1]}} & 8'hec)^
({{8{data_in[90][2]}} & 8'hf3)^
({{8{data_in[90][3]}} & 8'hcd)^
({{8{data_in[90][4]}} & 8'hb1)^
({{8{data_in[90][5]}} & 8'h49)^
({{8{data_in[90][6]}} & 8'h92)^
({{8{data_in[90][7]}} & 8'hf)^
({{8{data_in[91][0]}} & 8'h72)^
({{8{data_in[91][1]}} & 8'he4)^
({{8{data_in[91][2]}} & 8'he3)^
({{8{data_in[91][3]}} & 8'hed)^
({{8{data_in[91][4]}} & 8'hf1)^
({{8{data_in[91][5]}} & 8'hc9)^
({{8{data_in[91][6]}} & 8'hb9)^
({{8{data_in[91][7]}} & 8'h59)^
({{8{data_in[92][0]}} & 8'hb3)^
({{8{data_in[92][1]}} & 8'h4d)^
({{8{data_in[92][2]}} & 8'h9a)^
({{8{data_in[92][3]}} & 8'h1f)^
({{8{data_in[92][4]}} & 8'h3e)^
({{8{data_in[92][5]}} & 8'h7c)^
({{8{data_in[92][6]}} & 8'hf8)^
({{8{data_in[92][7]}} & 8'hdb)^
({{8{data_in[93][0]}} & 8'hf)^
({{8{data_in[93][1]}} & 8'h1e)^
({{8{data_in[93][2]}} & 8'h3c)^
({{8{data_in[93][3]}} & 8'h78)^
({{8{data_in[93][4]}} & 8'hf0)^
({{8{data_in[93][5]}} & 8'hcb)^
({{8{data_in[93][6]}} & 8'hbd)^
({{8{data_in[93][7]}} & 8'h51)^
({{8{data_in[94][0]}} & 8'hd2)^
({{8{data_in[94][1]}} & 8'h8f)^
({{8{data_in[94][2]}} & 8'h35)^
({{8{data_in[94][3]}} & 8'h6a)^
({{8{data_in[94][4]}} & 8'hd4)^
({{8{data_in[94][5]}} & 8'h83)^
({{8{data_in[94][6]}} & 8'h2d)^
({{8{data_in[94][7]}} & 8'h5a)^
({{8{data_in[95][0]}} & 8'h32)^
({{8{data_in[95][1]}} & 8'h64)^
({{8{data_in[95][2]}} & 8'hc8)^
({{8{data_in[95][3]}} & 8'hbb)^
({{8{data_in[95][4]}} & 8'h5d)^
({{8{data_in[95][5]}} & 8'hba)^
({{8{data_in[95][6]}} & 8'h5f)^
({{8{data_in[95][7]}} & 8'hbe)^
({{8{data_in[96][0]}} & 8'h65)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[96][1]}} & 8'hca)^
({{8{data_in[96][2]}} & 8'hbf)^
({{8{data_in[96][3]}} & 8'h55)^
({{8{data_in[96][4]}} & 8'haa)^
({{8{data_in[96][5]}} & 8'h7f)^
({{8{data_in[96][6]}} & 8'hfe)^
({{8{data_in[96][7]}} & 8'hd7)^
({{8{data_in[97][0]}} & 8'h3a)^
({{8{data_in[97][1]}} & 8'h74)^
({{8{data_in[97][2]}} & 8'he8)^
({{8{data_in[97][3]}} & 8'hfb)^
({{8{data_in[97][4]}} & 8'hdd)^
({{8{data_in[97][5]}} & 8'h91)^
({{8{data_in[97][6]}} & 8'h9)^
({{8{data_in[97][7]}} & 8'h12)^
({{8{data_in[98][0]}} & 8'hc0)^
({{8{data_in[98][1]}} & 8'hab)^
({{8{data_in[98][2]}} & 8'h7d)^
({{8{data_in[98][3]}} & 8'hfa)^
({{8{data_in[98][4]}} & 8'hdf)^
({{8{data_in[98][5]}} & 8'h95)^
({{8{data_in[98][6]}} & 8'h1)^
({{8{data_in[98][7]}} & 8'h2)^
({{8{data_in[99][0]}} & 8'hd8)^
({{8{data_in[99][1]}} & 8'h9b)^
({{8{data_in[99][2]}} & 8'h1d)^
({{8{data_in[99][3]}} & 8'h3a)^
({{8{data_in[99][4]}} & 8'h74)^
({{8{data_in[99][5]}} & 8'he8)^
({{8{data_in[99][6]}} & 8'hfb)^
({{8{data_in[99][7]}} & 8'hdd)^
({{8{data_in[100][0]}} & 8'h31)^
({{8{data_in[100][1]}} & 8'h62)^
({{8{data_in[100][2]}} & 8'hc4)^
({{8{data_in[100][3]}} & 8'ha3)^
({{8{data_in[100][4]}} & 8'h6d)^
({{8{data_in[100][5]}} & 8'hda)^
({{8{data_in[100][6]}} & 8'h9f)^
({{8{data_in[100][7]}} & 8'h15)^
({{8{data_in[101][0]}} & 8'hd2)^
({{8{data_in[101][1]}} & 8'h8f)^
({{8{data_in[101][2]}} & 8'h35)^
({{8{data_in[101][3]}} & 8'h6a)^
({{8{data_in[101][4]}} & 8'hd4)^
({{8{data_in[101][5]}} & 8'h83)^
({{8{data_in[101][6]}} & 8'h2d)^
({{8{data_in[101][7]}} & 8'h5a)^
({{8{data_in[102][0]}} & 8'hec)^
({{8{data_in[102][1]}} & 8'hf3)^
({{8{data_in[102][2]}} & 8'hcd)^
({{8{data_in[102][3]}} & 8'hb1)^
({{8{data_in[102][4]}} & 8'h49)^
({{8{data_in[102][5]}} & 8'h92)^
({{8{data_in[102][6]}} & 8'hf)^
({{8{data_in[102][7]}} & 8'h1e)^
({{8{data_in[103][0]}} & 8'hdc)^
({{8{data_in[103][1]}} & 8'h93)^
({{8{data_in[103][2]}} & 8'hd)^
({{8{data_in[103][3]}} & 8'h1a)^
({{8{data_in[103][4]}} & 8'h34)^
({{8{data_in[103][5]}} & 8'h68)^
({{8{data_in[103][6]}} & 8'hd0)^
({{8{data_in[103][7]}} & 8'h8b)^
({{8{data_in[104][0]}} & 8'ha0)^
({{8{data_in[104][1]}} & 8'h6b)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[104][2]}} & 8'h6d)^
({{8{data_in[104][3]}} & 8'h87)^
({{8{data_in[104][4]}} & 8'h25)^
({{8{data_in[104][5]}} & 8'h4a)^
({{8{data_in[104][6]}} & 8'h94)^
({{8{data_in[104][7]}} & 8'h3)^
({{8{data_in[105][0]}} & 8'h8b)^
({{8{data_in[105][1]}} & 8'h3d)^
({{8{data_in[105][2]}} & 8'h7a)^
({{8{data_in[105][3]}} & 8'hf4)^
({{8{data_in[105][4]}} & 8'hc3)^
({{8{data_in[105][5]}} & 8'had)^
({{8{data_in[105][6]}} & 8'h71)^
({{8{data_in[105][7]}} & 8'he2)^
({{8{data_in[106][0]}} & 8'h1f)^
({{8{data_in[106][1]}} & 8'h3e)^
({{8{data_in[106][2]}} & 8'h7c)^
({{8{data_in[106][3]}} & 8'hf8)^
({{8{data_in[106][4]}} & 8'hdb)^
({{8{data_in[106][5]}} & 8'h9d)^
({{8{data_in[106][6]}} & 8'h11)^
({{8{data_in[106][7]}} & 8'h22)^
({{8{data_in[107][0]}} & 8'hc4)^
({{8{data_in[107][1]}} & 8'ha3)^
({{8{data_in[107][2]}} & 8'h6d)^
({{8{data_in[107][3]}} & 8'hda)^
({{8{data_in[107][4]}} & 8'h9f)^
({{8{data_in[107][5]}} & 8'h15)^
({{8{data_in[107][6]}} & 8'h2a)^
({{8{data_in[107][7]}} & 8'h54)^
({{8{data_in[108][0]}} & 8'hae)^
({{8{data_in[108][1]}} & 8'h77)^
({{8{data_in[108][2]}} & 8'hee)^
({{8{data_in[108][3]}} & 8'hf7)^
({{8{data_in[108][4]}} & 8'hc5)^
({{8{data_in[108][5]}} & 8'ha1)^
({{8{data_in[108][6]}} & 8'h69)^
({{8{data_in[108][7]}} & 8'hd2)^
({{8{data_in[109][0]}} & 8'h70)^
({{8{data_in[109][1]}} & 8'he0)^
({{8{data_in[109][2]}} & 8'heb)^
({{8{data_in[109][3]}} & 8'hfd)^
({{8{data_in[109][4]}} & 8'hd1)^
({{8{data_in[109][5]}} & 8'h89)^
({{8{data_in[109][6]}} & 8'h39)^
({{8{data_in[109][7]}} & 8'h72)^
({{8{data_in[110][0]}} & 8'h3)^
({{8{data_in[110][1]}} & 8'h6)^
({{8{data_in[110][2]}} & 8'hc)^
({{8{data_in[110][3]}} & 8'h18)^
({{8{data_in[110][4]}} & 8'h30)^
({{8{data_in[110][5]}} & 8'h60)^
({{8{data_in[110][6]}} & 8'hc0)^
({{8{data_in[110][7]}} & 8'hab)^
({{8{data_in[111][0]}} & 8'h9d)^
({{8{data_in[111][1]}} & 8'h11)^
({{8{data_in[111][2]}} & 8'h22)^
({{8{data_in[111][3]}} & 8'h44)^
({{8{data_in[111][4]}} & 8'h88)^
({{8{data_in[111][5]}} & 8'h3b)^
({{8{data_in[111][6]}} & 8'h76)^
({{8{data_in[111][7]}} & 8'hec)^
({{8{data_in[112][0]}} & 8'h80)^
({{8{data_in[112][1]}} & 8'h2b)^
({{8{data_in[112][2]}} & 8'h56)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[112][3]}} & 8'hac)^
({{8{data_in[112][4]}} & 8'h73)^
({{8{data_in[112][5]}} & 8'he6)^
({{8{data_in[112][6]}} & 8'he7)^
({{8{data_in[112][7]}} & 8'he5)^
({{8{data_in[113][0]}} & 8'ha4)^
({{8{data_in[113][1]}} & 8'h63)^
({{8{data_in[113][2]}} & 8'hc6)^
({{8{data_in[113][3]}} & 8'ha7)^
({{8{data_in[113][4]}} & 8'h65)^
({{8{data_in[113][5]}} & 8'hca)^
({{8{data_in[113][6]}} & 8'hbf)^
({{8{data_in[113][7]}} & 8'h55)^
({{8{data_in[114][0]}} & 8'hbe)^
({{8{data_in[114][1]}} & 8'h57)^
({{8{data_in[114][2]}} & 8'hae)^
({{8{data_in[114][3]}} & 8'h77)^
({{8{data_in[114][4]}} & 8'hee)^
({{8{data_in[114][5]}} & 8'hf7)^
({{8{data_in[114][6]}} & 8'hc5)^
({{8{data_in[114][7]}} & 8'ha1)^
({{8{data_in[115][0]}} & 8'h56)^
({{8{data_in[115][1]}} & 8'hac)^
({{8{data_in[115][2]}} & 8'h73)^
({{8{data_in[115][3]}} & 8'he6)^
({{8{data_in[115][4]}} & 8'he7)^
({{8{data_in[115][5]}} & 8'he5)^
({{8{data_in[115][6]}} & 8'he1)^
({{8{data_in[115][7]}} & 8'he9)^
({{8{data_in[116][0]}} & 8'h49)^
({{8{data_in[116][1]}} & 8'h92)^
({{8{data_in[116][2]}} & 8'hf)^
({{8{data_in[116][3]}} & 8'h1e)^
({{8{data_in[116][4]}} & 8'h3c)^
({{8{data_in[116][5]}} & 8'h78)^
({{8{data_in[116][6]}} & 8'hf0)^
({{8{data_in[116][7]}} & 8'hcb)^
({{8{data_in[117][0]}} & 8'h1)^
({{8{data_in[117][1]}} & 8'h2)^
({{8{data_in[117][2]}} & 8'h4)^
({{8{data_in[117][3]}} & 8'h8)^
({{8{data_in[117][4]}} & 8'h10)^
({{8{data_in[117][5]}} & 8'h20)^
({{8{data_in[117][6]}} & 8'h40)^
({{8{data_in[117][7]}} & 8'h80)^
({{8{data_in[118][0]}} & 8'h2)^
({{8{data_in[118][1]}} & 8'h4)^
({{8{data_in[118][2]}} & 8'h8)^
({{8{data_in[118][3]}} & 8'h10)^
({{8{data_in[118][4]}} & 8'h20)^
({{8{data_in[118][5]}} & 8'h40)^
({{8{data_in[118][6]}} & 8'h80)^
({{8{data_in[118][7]}} & 8'h2b)^
({{8{data_in[119][0]}} & 8'h92)^
({{8{data_in[119][1]}} & 8'hf)^
({{8{data_in[119][2]}} & 8'h1e)^
({{8{data_in[119][3]}} & 8'h3c)^
({{8{data_in[119][4]}} & 8'h78)^
({{8{data_in[119][5]}} & 8'hf0)^
({{8{data_in[119][6]}} & 8'hcb)^
({{8{data_in[119][7]}} & 8'hbd)^
({{8{data_in[120][0]}} & 8'ha1)^
({{8{data_in[120][1]}} & 8'h69)^
({{8{data_in[120][2]}} & 8'hd2)^
({{8{data_in[120][3]}} & 8'h8f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[120][4]}} & 8'h35)^
({{8{data_in[120][5]}} & 8'h6a)^
({{8{data_in[120][6]}} & 8'hd4)^
({{8{data_in[120][7]}} & 8'h83)^
({{8{data_in[121][0]}} & 8'h39)^
({{8{data_in[121][1]}} & 8'h72)^
({{8{data_in[121][2]}} & 8'he4)^
({{8{data_in[121][3]}} & 8'he3)^
({{8{data_in[121][4]}} & 8'hed)^
({{8{data_in[121][5]}} & 8'hf1)^
({{8{data_in[121][6]}} & 8'hc9)^
({{8{data_in[121][7]}} & 8'hb9)^
({{8{data_in[122][0]}} & 8'h66)^
({{8{data_in[122][1]}} & 8'hcc)^
({{8{data_in[122][2]}} & 8'hb3)^
({{8{data_in[122][3]}} & 8'h4d)^
({{8{data_in[122][4]}} & 8'h9a)^
({{8{data_in[122][5]}} & 8'h1f)^
({{8{data_in[122][6]}} & 8'h3e)^
({{8{data_in[122][7]}} & 8'h7c)^
({{8{data_in[123][0]}} & 8'h4e)^
({{8{data_in[123][1]}} & 8'h9c)^
({{8{data_in[123][2]}} & 8'h13)^
({{8{data_in[123][3]}} & 8'h26)^
({{8{data_in[123][4]}} & 8'h4c)^
({{8{data_in[123][5]}} & 8'h98)^
({{8{data_in[123][6]}} & 8'h1b)^
({{8{data_in[123][7]}} & 8'h36)^
({{8{data_in[124][0]}} & 8'h28)^
({{8{data_in[124][1]}} & 8'h50)^
({{8{data_in[124][2]}} & 8'ha0)^
({{8{data_in[124][3]}} & 8'h6b)^
({{8{data_in[124][4]}} & 8'hd6)^
({{8{data_in[124][5]}} & 8'h87)^
({{8{data_in[124][6]}} & 8'h25)^
({{8{data_in[124][7]}} & 8'h4a)^
({{8{data_in[125][0]}} & 8'h90)^
({{8{data_in[125][1]}} & 8'hb)^
({{8{data_in[125][2]}} & 8'h16)^
({{8{data_in[125][3]}} & 8'h2c)^
({{8{data_in[125][4]}} & 8'h58)^
({{8{data_in[125][5]}} & 8'hb0)^
({{8{data_in[125][6]}} & 8'h4b)^
({{8{data_in[125][7]}} & 8'h96)^
({{8{data_in[126][0]}} & 8'he8)^
({{8{data_in[126][1]}} & 8'hfb)^
({{8{data_in[126][2]}} & 8'hdd)^
({{8{data_in[126][3]}} & 8'h91)^
({{8{data_in[126][4]}} & 8'h9)^
({{8{data_in[126][5]}} & 8'h12)^
({{8{data_in[126][6]}} & 8'h24)^
({{8{data_in[126][7]}} & 8'h48)^
({{8{data_in[127][0]}} & 8'h36)^
({{8{data_in[127][1]}} & 8'h6c)^
({{8{data_in[127][2]}} & 8'hd8)^
({{8{data_in[127][3]}} & 8'h9b)^
({{8{data_in[127][4]}} & 8'h1d)^
({{8{data_in[127][5]}} & 8'h3a)^
({{8{data_in[127][6]}} & 8'h74)^
({{8{data_in[127][7]}} & 8'he8)^
({{8{data_in[128][0]}} & 8'h53)^
({{8{data_in[128][1]}} & 8'ha6)^
({{8{data_in[128][2]}} & 8'h67)^
({{8{data_in[128][3]}} & 8'hce)^
({{8{data_in[128][4]}} & 8'hb7)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[128][5]}} & 8'h45)^
({{8{data_in[128][6]}} & 8'h8a)^
({{8{data_in[128][7]}} & 8'h3f)^
({{8{data_in[129][0]}} & 8'h4b)^
({{8{data_in[129][1]}} & 8'h96)^
({{8{data_in[129][2]}} & 8'h7)^
({{8{data_in[129][3]}} & 8'he)^
({{8{data_in[129][4]}} & 8'h1c)^
({{8{data_in[129][5]}} & 8'h38)^
({{8{data_in[129][6]}} & 8'h70)^
({{8{data_in[129][7]}} & 8'he0)^
({{8{data_in[130][0]}} & 8'h42)^
({{8{data_in[130][1]}} & 8'h84)^
({{8{data_in[130][2]}} & 8'h23)^
({{8{data_in[130][3]}} & 8'h46)^
({{8{data_in[130][4]}} & 8'h8c)^
({{8{data_in[130][5]}} & 8'h33)^
({{8{data_in[130][6]}} & 8'h66)^
({{8{data_in[130][7]}} & 8'hcc)^
({{8{data_in[131][0]}} & 8'he0)^
({{8{data_in[131][1]}} & 8'heb)^
({{8{data_in[131][2]}} & 8'hfd)^
({{8{data_in[131][3]}} & 8'hd1)^
({{8{data_in[131][4]}} & 8'h89)^
({{8{data_in[131][5]}} & 8'h39)^
({{8{data_in[131][6]}} & 8'h72)^
({{8{data_in[131][7]}} & 8'he4)^
({{8{data_in[132][0]}} & 8'hf6)^
({{8{data_in[132][1]}} & 8'hc7)^
({{8{data_in[132][2]}} & 8'ha5)^
({{8{data_in[132][3]}} & 8'h61)^
({{8{data_in[132][4]}} & 8'hc2)^
({{8{data_in[132][5]}} & 8'haf)^
({{8{data_in[132][6]}} & 8'h75)^
({{8{data_in[132][7]}} & 8'hea)^
({{8{data_in[133][0]}} & 8'h34)^
({{8{data_in[133][1]}} & 8'h68)^
({{8{data_in[133][2]}} & 8'hd0)^
({{8{data_in[133][3]}} & 8'h8b)^
({{8{data_in[133][4]}} & 8'h3d)^
({{8{data_in[133][5]}} & 8'h7a)^
({{8{data_in[133][6]}} & 8'hf4)^
({{8{data_in[133][7]}} & 8'hc3)^
({{8{data_in[134][0]}} & 8'hca)^
({{8{data_in[134][1]}} & 8'hbf)^
({{8{data_in[134][2]}} & 8'h55)^
({{8{data_in[134][3]}} & 8'haa)^
({{8{data_in[134][4]}} & 8'h7f)^
({{8{data_in[134][5]}} & 8'hfe)^
({{8{data_in[134][6]}} & 8'hd7)^
({{8{data_in[134][7]}} & 8'h85)^
({{8{data_in[135][0]}} & 8'hb6)^
({{8{data_in[135][1]}} & 8'h47)^
({{8{data_in[135][2]}} & 8'h8e)^
({{8{data_in[135][3]}} & 8'h37)^
({{8{data_in[135][4]}} & 8'h6e)^
({{8{data_in[135][5]}} & 8'hdc)^
({{8{data_in[135][6]}} & 8'h93)^
({{8{data_in[135][7]}} & 8'hd)^
({{8{data_in[136][0]}} & 8'he4)^
({{8{data_in[136][1]}} & 8'he3)^
({{8{data_in[136][2]}} & 8'hed)^
({{8{data_in[136][3]}} & 8'hf1)^
({{8{data_in[136][4]}} & 8'hc9)^
({{8{data_in[136][5]}} & 8'hb9)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[136][6]}} & 8'h59)^
({{8{data_in[136][7]}} & 8'hb2)^
({{8{data_in[137][0]}} & 8'ha0)^
({{8{data_in[137][1]}} & 8'h6b)^
({{8{data_in[137][2]}} & 8'hd6)^
({{8{data_in[137][3]}} & 8'h87)^
({{8{data_in[137][4]}} & 8'h25)^
({{8{data_in[137][5]}} & 8'h4a)^
({{8{data_in[137][6]}} & 8'h94)^
({{8{data_in[137][7]}} & 8'h3)^
({{8{data_in[138][0]}} & 8'hae)^
({{8{data_in[138][1]}} & 8'h77)^
({{8{data_in[138][2]}} & 8'hee)^
({{8{data_in[138][3]}} & 8'hf7)^
({{8{data_in[138][4]}} & 8'hc5)^
({{8{data_in[138][5]}} & 8'ha1)^
({{8{data_in[138][6]}} & 8'h69)^
({{8{data_in[138][7]}} & 8'hd2)^
({{8{data_in[139][0]}} & 8'h80)^
({{8{data_in[139][1]}} & 8'h2b)^
({{8{data_in[139][2]}} & 8'h56)^
({{8{data_in[139][3]}} & 8'hac)^
({{8{data_in[139][4]}} & 8'h73)^
({{8{data_in[139][5]}} & 8'he6)^
({{8{data_in[139][6]}} & 8'he7)^
({{8{data_in[139][7]}} & 8'he5)^
({{8{data_in[140][0]}} & 8'he5)^
({{8{data_in[140][1]}} & 8'he1)^
({{8{data_in[140][2]}} & 8'he9)^
({{8{data_in[140][3]}} & 8'hf9)^
({{8{data_in[140][4]}} & 8'hd9)^
({{8{data_in[140][5]}} & 8'h99)^
({{8{data_in[140][6]}} & 8'h19)^
({{8{data_in[140][7]}} & 8'h32)^
({{8{data_in[141][0]}} & 8'he)^
({{8{data_in[141][1]}} & 8'h1c)^
({{8{data_in[141][2]}} & 8'h38)^
({{8{data_in[141][3]}} & 8'h70)^
({{8{data_in[141][4]}} & 8'he0)^
({{8{data_in[141][5]}} & 8'heb)^
({{8{data_in[141][6]}} & 8'hfd)^
({{8{data_in[141][7]}} & 8'hd1)^
({{8{data_in[142][0]}} & 8'h83)^
({{8{data_in[142][1]}} & 8'h2d)^
({{8{data_in[142][2]}} & 8'h5a)^
({{8{data_in[142][3]}} & 8'hb4)^
({{8{data_in[142][4]}} & 8'h43)^
({{8{data_in[142][5]}} & 8'h86)^
({{8{data_in[142][6]}} & 8'h27)^
({{8{data_in[142][7]}} & 8'h4e)^
({{8{data_in[143][0]}} & 8'h9e)^
({{8{data_in[143][1]}} & 8'h17)^
({{8{data_in[143][2]}} & 8'h2e)^
({{8{data_in[143][3]}} & 8'h5c)^
({{8{data_in[143][4]}} & 8'hb8)^
({{8{data_in[143][5]}} & 8'h5b)^
({{8{data_in[143][6]}} & 8'hb6)^
({{8{data_in[143][7]}} & 8'h47)^
({{8{data_in[144][0]}} & 8'hbd)^
({{8{data_in[144][1]}} & 8'h51)^
({{8{data_in[144][2]}} & 8'ha2)^
({{8{data_in[144][3]}} & 8'h6f)^
({{8{data_in[144][4]}} & 8'hde)^
({{8{data_in[144][5]}} & 8'h97)^
({{8{data_in[144][6]}} & 8'h5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[144][7]}} & 8'ha)^
({{8{data_in[145][0]}} & 8'h45)^
({{8{data_in[145][1]}} & 8'h8a)^
({{8{data_in[145][2]}} & 8'h3f)^
({{8{data_in[145][3]}} & 8'h7e)^
({{8{data_in[145][4]}} & 8'hfc)^
({{8{data_in[145][5]}} & 8'hd3)^
({{8{data_in[145][6]}} & 8'h8d)^
({{8{data_in[145][7]}} & 8'h31)^
({{8{data_in[146][0]}} & 8'hf3)^
({{8{data_in[146][1]}} & 8'hcd)^
({{8{data_in[146][2]}} & 8'hb1)^
({{8{data_in[146][3]}} & 8'h49)^
({{8{data_in[146][4]}} & 8'h92)^
({{8{data_in[146][5]}} & 8'hf)^
({{8{data_in[146][6]}} & 8'h1e)^
({{8{data_in[146][7]}} & 8'h3c)^
({{8{data_in[147][0]}} & 8'hf5)^
({{8{data_in[147][1]}} & 8'hc1)^
({{8{data_in[147][2]}} & 8'ha9)^
({{8{data_in[147][3]}} & 8'h79)^
({{8{data_in[147][4]}} & 8'hf2)^
({{8{data_in[147][5]}} & 8'hcf)^
({{8{data_in[147][6]}} & 8'hb5)^
({{8{data_in[147][7]}} & 8'h41)^
({{8{data_in[148][0]}} & 8'h91)^
({{8{data_in[148][1]}} & 8'h9)^
({{8{data_in[148][2]}} & 8'h12)^
({{8{data_in[148][3]}} & 8'h24)^
({{8{data_in[148][4]}} & 8'h48)^
({{8{data_in[148][5]}} & 8'h90)^
({{8{data_in[148][6]}} & 8'hb)^
({{8{data_in[148][7]}} & 8'h16)^
({{8{data_in[149][0]}} & 8'h36)^
({{8{data_in[149][1]}} & 8'h6c)^
({{8{data_in[149][2]}} & 8'hd8)^
({{8{data_in[149][3]}} & 8'h9b)^
({{8{data_in[149][4]}} & 8'h1d)^
({{8{data_in[149][5]}} & 8'h3a)^
({{8{data_in[149][6]}} & 8'h74)^
({{8{data_in[149][7]}} & 8'he8)^
({{8{data_in[150][0]}} & 8'h3f)^
({{8{data_in[150][1]}} & 8'h7e)^
({{8{data_in[150][2]}} & 8'hfc)^
({{8{data_in[150][3]}} & 8'hd3)^
({{8{data_in[150][4]}} & 8'h8d)^
({{8{data_in[150][5]}} & 8'h31)^
({{8{data_in[150][6]}} & 8'h62)^
({{8{data_in[150][7]}} & 8'hc4)^
({{8{data_in[151][0]}} & 8'h1c)^
({{8{data_in[151][1]}} & 8'h38)^
({{8{data_in[151][2]}} & 8'h70)^
({{8{data_in[151][3]}} & 8'he0)^
({{8{data_in[151][4]}} & 8'heb)^
({{8{data_in[151][5]}} & 8'hfd)^
({{8{data_in[151][6]}} & 8'hd1)^
({{8{data_in[151][7]}} & 8'h89)^
({{8{data_in[152][0]}} & 8'hde)^
({{8{data_in[152][1]}} & 8'h97)^
({{8{data_in[152][2]}} & 8'h5)^
({{8{data_in[152][3]}} & 8'ha)^
({{8{data_in[152][4]}} & 8'h14)^
({{8{data_in[152][5]}} & 8'h28)^
({{8{data_in[152][6]}} & 8'h50)^
({{8{data_in[152][7]}} & 8'ha0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[153][0]}} & 8'ha1)^
({{8{data_in[153][1]}} & 8'h69)^
({{8{data_in[153][2]}} & 8'hd2)^
({{8{data_in[153][3]}} & 8'h8f)^
({{8{data_in[153][4]}} & 8'h35)^
({{8{data_in[153][5]}} & 8'h6a)^
({{8{data_in[153][6]}} & 8'hd4)^
({{8{data_in[153][7]}} & 8'h83)^
({{8{data_in[154][0]}} & 8'h8d)^
({{8{data_in[154][1]}} & 8'h31)^
({{8{data_in[154][2]}} & 8'h62)^
({{8{data_in[154][3]}} & 8'hc4)^
({{8{data_in[154][4]}} & 8'ha3)^
({{8{data_in[154][5]}} & 8'h6d)^
({{8{data_in[154][6]}} & 8'hda)^
({{8{data_in[154][7]}} & 8'h9f)^
({{8{data_in[155][0]}} & 8'hb0)^
({{8{data_in[155][1]}} & 8'h4b)^
({{8{data_in[155][2]}} & 8'h96)^
({{8{data_in[155][3]}} & 8'h7)^
({{8{data_in[155][4]}} & 8'he)^
({{8{data_in[155][5]}} & 8'h1c)^
({{8{data_in[155][6]}} & 8'h38)^
({{8{data_in[155][7]}} & 8'h70)^
({{8{data_in[156][0]}} & 8'hdc)^
({{8{data_in[156][1]}} & 8'h93)^
({{8{data_in[156][2]}} & 8'hd)^
({{8{data_in[156][3]}} & 8'h1a)^
({{8{data_in[156][4]}} & 8'h34)^
({{8{data_in[156][5]}} & 8'h68)^
({{8{data_in[156][6]}} & 8'hd0)^
({{8{data_in[156][7]}} & 8'h8b)^
({{8{data_in[157][0]}} & 8'h48)^
({{8{data_in[157][1]}} & 8'h90)^
({{8{data_in[157][2]}} & 8'hb)^
({{8{data_in[157][3]}} & 8'h16)^
({{8{data_in[157][4]}} & 8'h2c)^
({{8{data_in[157][5]}} & 8'h58)^
({{8{data_in[157][6]}} & 8'hb0)^
({{8{data_in[157][7]}} & 8'h4b)^
({{8{data_in[158][0]}} & 8'hb7)^
({{8{data_in[158][1]}} & 8'h45)^
({{8{data_in[158][2]}} & 8'h8a)^
({{8{data_in[158][3]}} & 8'h3f)^
({{8{data_in[158][4]}} & 8'h7e)^
({{8{data_in[158][5]}} & 8'hfc)^
({{8{data_in[158][6]}} & 8'hd3)^
({{8{data_in[158][7]}} & 8'h8d)^
({{8{data_in[159][0]}} & 8'h71)^
({{8{data_in[159][1]}} & 8'he2)^
({{8{data_in[159][2]}} & 8'hef)^
({{8{data_in[159][3]}} & 8'hf5)^
({{8{data_in[159][4]}} & 8'hc1)^
({{8{data_in[159][5]}} & 8'ha9)^
({{8{data_in[159][6]}} & 8'h79)^
({{8{data_in[159][7]}} & 8'hf2)^
({{8{data_in[160][0]}} & 8'h35)^
({{8{data_in[160][1]}} & 8'h6a)^
({{8{data_in[160][2]}} & 8'hd4)^
({{8{data_in[160][3]}} & 8'h83)^
({{8{data_in[160][4]}} & 8'h2d)^
({{8{data_in[160][5]}} & 8'h5a)^
({{8{data_in[160][6]}} & 8'hb4)^
({{8{data_in[160][7]}} & 8'h43)^
({{8{data_in[161][0]}} & 8'h25)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[161][1]}} & 8'h4a)^
({{8{data_in[161][2]}} & 8'h94)^
({{8{data_in[161][3]}} & 8'h3)^
({{8{data_in[161][4]}} & 8'h6)^
({{8{data_in[161][5]}} & 8'hc)^
({{8{data_in[161][6]}} & 8'h18)^
({{8{data_in[161][7]}} & 8'h30)^
({{8{data_in[162][0]}} & 8'h51)^
({{8{data_in[162][1]}} & 8'ha2)^
({{8{data_in[162][2]}} & 8'h6f)^
({{8{data_in[162][3]}} & 8'hde)^
({{8{data_in[162][4]}} & 8'h97)^
({{8{data_in[162][5]}} & 8'h5)^
({{8{data_in[162][6]}} & 8'ha)^
({{8{data_in[162][7]}} & 8'h14)^
({{8{data_in[163][0]}} & 8'hfa)^
({{8{data_in[163][1]}} & 8'hdf)^
({{8{data_in[163][2]}} & 8'h95)^
({{8{data_in[163][3]}} & 8'h1)^
({{8{data_in[163][4]}} & 8'h2)^
({{8{data_in[163][5]}} & 8'h4)^
({{8{data_in[163][6]}} & 8'h8)^
({{8{data_in[163][7]}} & 8'h10)^
({{8{data_in[164][0]}} & 8'hcb)^
({{8{data_in[164][1]}} & 8'hbd)^
({{8{data_in[164][2]}} & 8'h51)^
({{8{data_in[164][3]}} & 8'ha2)^
({{8{data_in[164][4]}} & 8'h6f)^
({{8{data_in[164][5]}} & 8'hde)^
({{8{data_in[164][6]}} & 8'h97)^
({{8{data_in[164][7]}} & 8'h5)^
({{8{data_in[165][0]}} & 8'h2c)^
({{8{data_in[165][1]}} & 8'h58)^
({{8{data_in[165][2]}} & 8'hb0)^
({{8{data_in[165][3]}} & 8'h4b)^
({{8{data_in[165][4]}} & 8'h96)^
({{8{data_in[165][5]}} & 8'h7)^
({{8{data_in[165][6]}} & 8'he)^
({{8{data_in[165][7]}} & 8'h1c)^
({{8{data_in[166][0]}} & 8'h27)^
({{8{data_in[166][1]}} & 8'h4e)^
({{8{data_in[166][2]}} & 8'h9c)^
({{8{data_in[166][3]}} & 8'h13)^
({{8{data_in[166][4]}} & 8'h26)^
({{8{data_in[166][5]}} & 8'h4c)^
({{8{data_in[166][6]}} & 8'h98)^
({{8{data_in[166][7]}} & 8'h1b)^
({{8{data_in[167][0]}} & 8'h68)^
({{8{data_in[167][1]}} & 8'hd0)^
({{8{data_in[167][2]}} & 8'h8b)^
({{8{data_in[167][3]}} & 8'h3d)^
({{8{data_in[167][4]}} & 8'h7a)^
({{8{data_in[167][5]}} & 8'hf4)^
({{8{data_in[167][6]}} & 8'hc3)^
({{8{data_in[167][7]}} & 8'had)^
({{8{data_in[168][0]}} & 8'h5a)^
({{8{data_in[168][1]}} & 8'hb4)^
({{8{data_in[168][2]}} & 8'h43)^
({{8{data_in[168][3]}} & 8'h86)^
({{8{data_in[168][4]}} & 8'h27)^
({{8{data_in[168][5]}} & 8'h4e)^
({{8{data_in[168][6]}} & 8'h9c)^
({{8{data_in[168][7]}} & 8'h13)^
({{8{data_in[169][0]}} & 8'hb2)^
({{8{data_in[169][1]}} & 8'h4f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[169][2]}} & 8'h9e)^
({{8{data_in[169][3]}} & 8'h17)^
({{8{data_in[169][4]}} & 8'h2e)^
({{8{data_in[169][5]}} & 8'h5c)^
({{8{data_in[169][6]}} & 8'hb8)^
({{8{data_in[169][7]}} & 8'h5b)^
({{8{data_in[170][0]}} & 8'h4f)^
({{8{data_in[170][1]}} & 8'h9e)^
({{8{data_in[170][2]}} & 8'h17)^
({{8{data_in[170][3]}} & 8'h2e)^
({{8{data_in[170][4]}} & 8'h5c)^
({{8{data_in[170][5]}} & 8'hb8)^
({{8{data_in[170][6]}} & 8'h5b)^
({{8{data_in[170][7]}} & 8'hb6)^
({{8{data_in[171][0]}} & 8'h9a)^
({{8{data_in[171][1]}} & 8'h1f)^
({{8{data_in[171][2]}} & 8'h3e)^
({{8{data_in[171][3]}} & 8'h7c)^
({{8{data_in[171][4]}} & 8'hf8)^
({{8{data_in[171][5]}} & 8'hdb)^
({{8{data_in[171][6]}} & 8'h9d)^
({{8{data_in[171][7]}} & 8'h11)^
({{8{data_in[172][0]}} & 8'h9c)^
({{8{data_in[172][1]}} & 8'h13)^
({{8{data_in[172][2]}} & 8'h26)^
({{8{data_in[172][3]}} & 8'h4c)^
({{8{data_in[172][4]}} & 8'h98)^
({{8{data_in[172][5]}} & 8'h1b)^
({{8{data_in[172][6]}} & 8'h36)^
({{8{data_in[172][7]}} & 8'h6c)^
({{8{data_in[173][0]}} & 8'h2e)^
({{8{data_in[173][1]}} & 8'h5c)^
({{8{data_in[173][2]}} & 8'hb8)^
({{8{data_in[173][3]}} & 8'h5b)^
({{8{data_in[173][4]}} & 8'hb6)^
({{8{data_in[173][5]}} & 8'h47)^
({{8{data_in[173][6]}} & 8'h8e)^
({{8{data_in[173][7]}} & 8'h37)^
({{8{data_in[174][0]}} & 8'hb5)^
({{8{data_in[174][1]}} & 8'h41)^
({{8{data_in[174][2]}} & 8'h82)^
({{8{data_in[174][3]}} & 8'h2f)^
({{8{data_in[174][4]}} & 8'h5e)^
({{8{data_in[174][5]}} & 8'hbc)^
({{8{data_in[174][6]}} & 8'h53)^
({{8{data_in[174][7]}} & 8'ha6)^
({{8{data_in[175][0]}} & 8'h7b)^
({{8{data_in[175][1]}} & 8'hf6)^
({{8{data_in[175][2]}} & 8'hc7)^
({{8{data_in[175][3]}} & 8'ha5)^
({{8{data_in[175][4]}} & 8'h61)^
({{8{data_in[175][5]}} & 8'hc2)^
({{8{data_in[175][6]}} & 8'haf)^
({{8{data_in[175][7]}} & 8'h75)^
({{8{data_in[176][0]}} & 8'h9e)^
({{8{data_in[176][1]}} & 8'h17)^
({{8{data_in[176][2]}} & 8'h2e)^
({{8{data_in[176][3]}} & 8'h5c)^
({{8{data_in[176][4]}} & 8'hb8)^
({{8{data_in[176][5]}} & 8'h5b)^
({{8{data_in[176][6]}} & 8'hb6)^
({{8{data_in[176][7]}} & 8'h47)^
({{8{data_in[177][0]}} & 8'hec)^
({{8{data_in[177][1]}} & 8'hf3)^
({{8{data_in[177][2]}} & 8'hcd)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[177][3]}} & 8'hb1)^
({{8{data_in[177][4]}} & 8'h49)^
({{8{data_in[177][5]}} & 8'h92)^
({{8{data_in[177][6]}} & 8'hf)^
({{8{data_in[177][7]}} & 8'h1e)^
({{8{data_in[178][0]}} & 8'h44)^
({{8{data_in[178][1]}} & 8'h88)^
({{8{data_in[178][2]}} & 8'h3b)^
({{8{data_in[178][3]}} & 8'h76)^
({{8{data_in[178][4]}} & 8'hec)^
({{8{data_in[178][5]}} & 8'hf3)^
({{8{data_in[178][6]}} & 8'hcd)^
({{8{data_in[178][7]}} & 8'hb1)^
({{8{data_in[179][0]}} & 8'he3)^
({{8{data_in[179][1]}} & 8'hed)^
({{8{data_in[179][2]}} & 8'hf1)^
({{8{data_in[179][3]}} & 8'hc9)^
({{8{data_in[179][4]}} & 8'hb9)^
({{8{data_in[179][5]}} & 8'h59)^
({{8{data_in[179][6]}} & 8'hb2)^
({{8{data_in[179][7]}} & 8'h4f)^
({{8{data_in[180][0]}} & 8'he4)^
({{8{data_in[180][1]}} & 8'he3)^
({{8{data_in[180][2]}} & 8'hed)^
({{8{data_in[180][3]}} & 8'hf1)^
({{8{data_in[180][4]}} & 8'hc9)^
({{8{data_in[180][5]}} & 8'hb9)^
({{8{data_in[180][6]}} & 8'h59)^
({{8{data_in[180][7]}} & 8'hb2)^
({{8{data_in[181][0]}} & 8'h60)^
({{8{data_in[181][1]}} & 8'hc0)^
({{8{data_in[181][2]}} & 8'hab)^
({{8{data_in[181][3]}} & 8'h7d)^
({{8{data_in[181][4]}} & 8'hfa)^
({{8{data_in[181][5]}} & 8'hdf)^
({{8{data_in[181][6]}} & 8'h95)^
({{8{data_in[181][7]}} & 8'h1)^
({{8{data_in[182][0]}} & 8'h6b)^
({{8{data_in[182][1]}} & 8'hd6)^
({{8{data_in[182][2]}} & 8'h87)^
({{8{data_in[182][3]}} & 8'h25)^
({{8{data_in[182][4]}} & 8'h4a)^
({{8{data_in[182][5]}} & 8'h94)^
({{8{data_in[182][6]}} & 8'h3)^
({{8{data_in[182][7]}} & 8'h6)^
({{8{data_in[183][0]}} & 8'hd2)^
({{8{data_in[183][1]}} & 8'h8f)^
({{8{data_in[183][2]}} & 8'h35)^
({{8{data_in[183][3]}} & 8'h6a)^
({{8{data_in[183][4]}} & 8'hd4)^
({{8{data_in[183][5]}} & 8'h83)^
({{8{data_in[183][6]}} & 8'h2d)^
({{8{data_in[183][7]}} & 8'h5a)^
({{8{data_in[184][0]}} & 8'hf)^
({{8{data_in[184][1]}} & 8'h1e)^
({{8{data_in[184][2]}} & 8'h3c)^
({{8{data_in[184][3]}} & 8'h78)^
({{8{data_in[184][4]}} & 8'hf0)^
({{8{data_in[184][5]}} & 8'hcb)^
({{8{data_in[184][6]}} & 8'hbd)^
({{8{data_in[184][7]}} & 8'h51)^
({{8{data_in[185][0]}} & 8'h37)^
({{8{data_in[185][1]}} & 8'h6e)^
({{8{data_in[185][2]}} & 8'hdc)^
({{8{data_in[185][3]}} & 8'h93)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[185][4]}} & 8'hd)^
({{8{data_in[185][5]}} & 8'h1a)^
({{8{data_in[185][6]}} & 8'h34)^
({{8{data_in[185][7]}} & 8'h68)^
({{8{data_in[186][0]}} & 8'hc6)^
({{8{data_in[186][1]}} & 8'ha7)^
({{8{data_in[186][2]}} & 8'h65)^
({{8{data_in[186][3]}} & 8'hca)^
({{8{data_in[186][4]}} & 8'hbf)^
({{8{data_in[186][5]}} & 8'h55)^
({{8{data_in[186][6]}} & 8'haa)^
({{8{data_in[186][7]}} & 8'h7f)^
({{8{data_in[187][0]}} & 8'h8d)^
({{8{data_in[187][1]}} & 8'h31)^
({{8{data_in[187][2]}} & 8'h62)^
({{8{data_in[187][3]}} & 8'hc4)^
({{8{data_in[187][4]}} & 8'ha3)^
({{8{data_in[187][5]}} & 8'h6d)^
({{8{data_in[187][6]}} & 8'hda)^
({{8{data_in[187][7]}} & 8'h9f)^
({{8{data_in[188][0]}} & 8'h82)^
({{8{data_in[188][1]}} & 8'h2f)^
({{8{data_in[188][2]}} & 8'h5e)^
({{8{data_in[188][3]}} & 8'hbc)^
({{8{data_in[188][4]}} & 8'h53)^
({{8{data_in[188][5]}} & 8'ha6)^
({{8{data_in[188][6]}} & 8'h67)^
({{8{data_in[188][7]}} & 8'hce)^
({{8{data_in[189][0]}} & 8'h4e)^
({{8{data_in[189][1]}} & 8'h9c)^
({{8{data_in[189][2]}} & 8'h13)^
({{8{data_in[189][3]}} & 8'h26)^
({{8{data_in[189][4]}} & 8'h4c)^
({{8{data_in[189][5]}} & 8'h98)^
({{8{data_in[189][6]}} & 8'h1b)^
({{8{data_in[189][7]}} & 8'h36)^
({{8{data_in[190][0]}} & 8'hb5)^
({{8{data_in[190][1]}} & 8'h41)^
({{8{data_in[190][2]}} & 8'h82)^
({{8{data_in[190][3]}} & 8'h2f)^
({{8{data_in[190][4]}} & 8'h5e)^
({{8{data_in[190][5]}} & 8'hbc)^
({{8{data_in[190][6]}} & 8'h53)^
({{8{data_in[190][7]}} & 8'ha6)^
({{8{data_in[191][0]}} & 8'h43)^
({{8{data_in[191][1]}} & 8'h86)^
({{8{data_in[191][2]}} & 8'h27)^
({{8{data_in[191][3]}} & 8'h4e)^
({{8{data_in[191][4]}} & 8'h9c)^
({{8{data_in[191][5]}} & 8'h13)^
({{8{data_in[191][6]}} & 8'h26)^
({{8{data_in[191][7]}} & 8'h4c)^
({{8{data_in[192][0]}} & 8'h6f)^
({{8{data_in[192][1]}} & 8'hde)^
({{8{data_in[192][2]}} & 8'h97)^
({{8{data_in[192][3]}} & 8'h5)^
({{8{data_in[192][4]}} & 8'ha)^
({{8{data_in[192][5]}} & 8'h14)^
({{8{data_in[192][6]}} & 8'h28)^
({{8{data_in[192][7]}} & 8'h50)^
({{8{data_in[193][0]}} & 8'h73)^
({{8{data_in[193][1]}} & 8'he6)^
({{8{data_in[193][2]}} & 8'he7)^
({{8{data_in[193][3]}} & 8'he5)^
({{8{data_in[193][4]}} & 8'he1)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[193][5]}} & 8'he9)^
({{8{data_in[193][6]}} & 8'hf9)^
({{8{data_in[193][7]}} & 8'hd9)^
({{8{data_in[194][0]}} & 8'hbe)^
({{8{data_in[194][1]}} & 8'h57)^
({{8{data_in[194][2]}} & 8'hae)^
({{8{data_in[194][3]}} & 8'h77)^
({{8{data_in[194][4]}} & 8'hee)^
({{8{data_in[194][5]}} & 8'hf7)^
({{8{data_in[194][6]}} & 8'hc5)^
({{8{data_in[194][7]}} & 8'ha1)^
({{8{data_in[195][0]}} & 8'h27)^
({{8{data_in[195][1]}} & 8'h4e)^
({{8{data_in[195][2]}} & 8'h9c)^
({{8{data_in[195][3]}} & 8'h13)^
({{8{data_in[195][4]}} & 8'h26)^
({{8{data_in[195][5]}} & 8'h4c)^
({{8{data_in[195][6]}} & 8'h98)^
({{8{data_in[195][7]}} & 8'h1b)^
({{8{data_in[196][0]}} & 8'hb3)^
({{8{data_in[196][1]}} & 8'h4d)^
({{8{data_in[196][2]}} & 8'h9a)^
({{8{data_in[196][3]}} & 8'h1f)^
({{8{data_in[196][4]}} & 8'h3e)^
({{8{data_in[196][5]}} & 8'h7c)^
({{8{data_in[196][6]}} & 8'hf8)^
({{8{data_in[196][7]}} & 8'hdb)^
({{8{data_in[197][0]}} & 8'h60)^
({{8{data_in[197][1]}} & 8'hc0)^
({{8{data_in[197][2]}} & 8'hab)^
({{8{data_in[197][3]}} & 8'h7d)^
({{8{data_in[197][4]}} & 8'hfa)^
({{8{data_in[197][5]}} & 8'hdf)^
({{8{data_in[197][6]}} & 8'h95)^
({{8{data_in[197][7]}} & 8'h1)^
({{8{data_in[198][0]}} & 8'hda)^
({{8{data_in[198][1]}} & 8'h9f)^
({{8{data_in[198][2]}} & 8'h15)^
({{8{data_in[198][3]}} & 8'h2a)^
({{8{data_in[198][4]}} & 8'h54)^
({{8{data_in[198][5]}} & 8'ha8)^
({{8{data_in[198][6]}} & 8'h7b)^
({{8{data_in[198][7]}} & 8'hf6)^
({{8{data_in[199][0]}} & 8'h22)^
({{8{data_in[199][1]}} & 8'h44)^
({{8{data_in[199][2]}} & 8'h88)^
({{8{data_in[199][3]}} & 8'h3b)^
({{8{data_in[199][4]}} & 8'h76)^
({{8{data_in[199][5]}} & 8'hec)^
({{8{data_in[199][6]}} & 8'hf3)^
({{8{data_in[199][7]}} & 8'hcd)^
({{8{data_in[200][0]}} & 8'hf8)^
({{8{data_in[200][1]}} & 8'hdb)^
({{8{data_in[200][2]}} & 8'h9d)^
({{8{data_in[200][3]}} & 8'h11)^
({{8{data_in[200][4]}} & 8'h22)^
({{8{data_in[200][5]}} & 8'h44)^
({{8{data_in[200][6]}} & 8'h88)^
({{8{data_in[200][7]}} & 8'h3b)^
({{8{data_in[201][0]}} & 8'h48)^
({{8{data_in[201][1]}} & 8'h90)^
({{8{data_in[201][2]}} & 8'hb)^
({{8{data_in[201][3]}} & 8'h16)^
({{8{data_in[201][4]}} & 8'h2c)^
({{8{data_in[201][5]}} & 8'h58)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[201][6]}} & 8'hb0)^
({{8{data_in[201][7]}} & 8'h4b)^
({{8{data_in[202][0]}} & 8'h36)^
({{8{data_in[202][1]}} & 8'h6c)^
({{8{data_in[202][2]}} & 8'hd8)^
({{8{data_in[202][3]}} & 8'h9b)^
({{8{data_in[202][4]}} & 8'h1d)^
({{8{data_in[202][5]}} & 8'h3a)^
({{8{data_in[202][6]}} & 8'h74)^
({{8{data_in[202][7]}} & 8'he8)^
({{8{data_in[203][0]}} & 8'hf3)^
({{8{data_in[203][1]}} & 8'hcd)^
({{8{data_in[203][2]}} & 8'hb1)^
({{8{data_in[203][3]}} & 8'h49)^
({{8{data_in[203][4]}} & 8'h92)^
({{8{data_in[203][5]}} & 8'hf)^
({{8{data_in[203][6]}} & 8'h1e)^
({{8{data_in[203][7]}} & 8'h3c)^
({{8{data_in[204][0]}} & 8'h7b)^
({{8{data_in[204][1]}} & 8'hf6)^
({{8{data_in[204][2]}} & 8'hc7)^
({{8{data_in[204][3]}} & 8'ha5)^
({{8{data_in[204][4]}} & 8'h61)^
({{8{data_in[204][5]}} & 8'hc2)^
({{8{data_in[204][6]}} & 8'haf)^
({{8{data_in[204][7]}} & 8'h75)^
({{8{data_in[205][0]}} & 8'h68)^
({{8{data_in[205][1]}} & 8'hd0)^
({{8{data_in[205][2]}} & 8'h8b)^
({{8{data_in[205][3]}} & 8'h3d)^
({{8{data_in[205][4]}} & 8'h7a)^
({{8{data_in[205][5]}} & 8'hf4)^
({{8{data_in[205][6]}} & 8'hc3)^
({{8{data_in[205][7]}} & 8'had)^
({{8{data_in[206][0]}} & 8'h39)^
({{8{data_in[206][1]}} & 8'h72)^
({{8{data_in[206][2]}} & 8'he4)^
({{8{data_in[206][3]}} & 8'he3)^
({{8{data_in[206][4]}} & 8'hed)^
({{8{data_in[206][5]}} & 8'hf1)^
({{8{data_in[206][6]}} & 8'hc9)^
({{8{data_in[206][7]}} & 8'hb9)^
({{8{data_in[207][0]}} & 8'h7f)^
({{8{data_in[207][1]}} & 8'hfe)^
({{8{data_in[207][2]}} & 8'hd7)^
({{8{data_in[207][3]}} & 8'h85)^
({{8{data_in[207][4]}} & 8'h21)^
({{8{data_in[207][5]}} & 8'h42)^
({{8{data_in[207][6]}} & 8'h84)^
({{8{data_in[207][7]}} & 8'h23)^
({{8{data_in[208][0]}} & 8'h24)^
({{8{data_in[208][1]}} & 8'h48)^
({{8{data_in[208][2]}} & 8'h90)^
({{8{data_in[208][3]}} & 8'hb)^
({{8{data_in[208][4]}} & 8'h16)^
({{8{data_in[208][5]}} & 8'h2c)^
({{8{data_in[208][6]}} & 8'h58)^
({{8{data_in[208][7]}} & 8'hb0)^
({{8{data_in[209][0]}} & 8'h91)^
({{8{data_in[209][1]}} & 8'h9)^
({{8{data_in[209][2]}} & 8'h12)^
({{8{data_in[209][3]}} & 8'h24)^
({{8{data_in[209][4]}} & 8'h48)^
({{8{data_in[209][5]}} & 8'h90)^
({{8{data_in[209][6]}} & 8'hb)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[209][7]}} & 8'h16)^
({{8{data_in[210][0]}} & 8'haf)^
({{8{data_in[210][1]}} & 8'h75)^
({{8{data_in[210][2]}} & 8'hea)^
({{8{data_in[210][3]}} & 8'hff)^
({{8{data_in[210][4]}} & 8'hd5)^
({{8{data_in[210][5]}} & 8'h81)^
({{8{data_in[210][6]}} & 8'h29)^
({{8{data_in[210][7]}} & 8'h52)^
({{8{data_in[211][0]}} & 8'h83)^
({{8{data_in[211][1]}} & 8'h2d)^
({{8{data_in[211][2]}} & 8'h5a)^
({{8{data_in[211][3]}} & 8'hb4)^
({{8{data_in[211][4]}} & 8'h43)^
({{8{data_in[211][5]}} & 8'h86)^
({{8{data_in[211][6]}} & 8'h27)^
({{8{data_in[211][7]}} & 8'h4e)^
({{8{data_in[212][0]}} & 8'ha0)^
({{8{data_in[212][1]}} & 8'h6b)^
({{8{data_in[212][2]}} & 8'hd6)^
({{8{data_in[212][3]}} & 8'h87)^
({{8{data_in[212][4]}} & 8'h25)^
({{8{data_in[212][5]}} & 8'h4a)^
({{8{data_in[212][6]}} & 8'h94)^
({{8{data_in[212][7]}} & 8'h3)^
({{8{data_in[213][0]}} & 8'hc6)^
({{8{data_in[213][1]}} & 8'ha7)^
({{8{data_in[213][2]}} & 8'h65)^
({{8{data_in[213][3]}} & 8'hca)^
({{8{data_in[213][4]}} & 8'hbf)^
({{8{data_in[213][5]}} & 8'h55)^
({{8{data_in[213][6]}} & 8'haa)^
({{8{data_in[213][7]}} & 8'h7f)^
({{8{data_in[214][0]}} & 8'h14)^
({{8{data_in[214][1]}} & 8'h28)^
({{8{data_in[214][2]}} & 8'h50)^
({{8{data_in[214][3]}} & 8'ha0)^
({{8{data_in[214][4]}} & 8'h6b)^
({{8{data_in[214][5]}} & 8'hd6)^
({{8{data_in[214][6]}} & 8'h87)^
({{8{data_in[214][7]}} & 8'h25)^
({{8{data_in[215][0]}} & 8'hea)^
({{8{data_in[215][1]}} & 8'hff)^
({{8{data_in[215][2]}} & 8'hd5)^
({{8{data_in[215][3]}} & 8'h81)^
({{8{data_in[215][4]}} & 8'h29)^
({{8{data_in[215][5]}} & 8'h52)^
({{8{data_in[215][6]}} & 8'ha4)^
({{8{data_in[215][7]}} & 8'h63)^
({{8{data_in[216][0]}} & 8'h44)^
({{8{data_in[216][1]}} & 8'h88)^
({{8{data_in[216][2]}} & 8'h3b)^
({{8{data_in[216][3]}} & 8'h76)^
({{8{data_in[216][4]}} & 8'hec)^
({{8{data_in[216][5]}} & 8'hf3)^
({{8{data_in[216][6]}} & 8'hcd)^
({{8{data_in[216][7]}} & 8'hb1)^
({{8{data_in[217][0]}} & 8'h23)^
({{8{data_in[217][1]}} & 8'h46)^
({{8{data_in[217][2]}} & 8'h8c)^
({{8{data_in[217][3]}} & 8'h33)^
({{8{data_in[217][4]}} & 8'h66)^
({{8{data_in[217][5]}} & 8'hcc)^
({{8{data_in[217][6]}} & 8'hb3)^
({{8{data_in[217][7]}} & 8'h4d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[218][0]}} & 8'hab)^
({{8{data_in[218][1]}} & 8'h7d)^
({{8{data_in[218][2]}} & 8'hfa)^
({{8{data_in[218][3]}} & 8'hdf)^
({{8{data_in[218][4]}} & 8'h95)^
({{8{data_in[218][5]}} & 8'h1)^
({{8{data_in[218][6]}} & 8'h2)^
({{8{data_in[218][7]}} & 8'h4)^
({{8{data_in[219][0]}} & 8'hf3)^
({{8{data_in[219][1]}} & 8'hcd)^
({{8{data_in[219][2]}} & 8'hb1)^
({{8{data_in[219][3]}} & 8'h49)^
({{8{data_in[219][4]}} & 8'h92)^
({{8{data_in[219][5]}} & 8'hf)^
({{8{data_in[219][6]}} & 8'h1e)^
({{8{data_in[219][7]}} & 8'h3c)^
({{8{data_in[220][0]}} & 8'he1)^
({{8{data_in[220][1]}} & 8'he9)^
({{8{data_in[220][2]}} & 8'hf9)^
({{8{data_in[220][3]}} & 8'hd9)^
({{8{data_in[220][4]}} & 8'h99)^
({{8{data_in[220][5]}} & 8'h19)^
({{8{data_in[220][6]}} & 8'h32)^
({{8{data_in[220][7]}} & 8'h64)^
({{8{data_in[221][0]}} & 8'hd8)^
({{8{data_in[221][1]}} & 8'h9b)^
({{8{data_in[221][2]}} & 8'h1d)^
({{8{data_in[221][3]}} & 8'h3a)^
({{8{data_in[221][4]}} & 8'h74)^
({{8{data_in[221][5]}} & 8'he8)^
({{8{data_in[221][6]}} & 8'fib)^
({{8{data_in[221][7]}} & 8'hdd)^
({{8{data_in[222][0]}} & 8'h83)^
({{8{data_in[222][1]}} & 8'h2d)^
({{8{data_in[222][2]}} & 8'h5a)^
({{8{data_in[222][3]}} & 8'hb4)^
({{8{data_in[222][4]}} & 8'h43)^
({{8{data_in[222][5]}} & 8'h86)^
({{8{data_in[222][6]}} & 8'h27)^
({{8{data_in[222][7]}} & 8'h4e)^
({{8{data_in[223][0]}} & 8'ha1)^
({{8{data_in[223][1]}} & 8'h69)^
({{8{data_in[223][2]}} & 8'hd2)^
({{8{data_in[223][3]}} & 8'h8f)^
({{8{data_in[223][4]}} & 8'h35)^
({{8{data_in[223][5]}} & 8'h6a)^
({{8{data_in[223][6]}} & 8'hd4)^
({{8{data_in[223][7]}} & 8'h83)^
({{8{data_in[224][0]}} & 8'h95)^
({{8{data_in[224][1]}} & 8'h1)^
({{8{data_in[224][2]}} & 8'h2)^
({{8{data_in[224][3]}} & 8'h4)^
({{8{data_in[224][4]}} & 8'h8)^
({{8{data_in[224][5]}} & 8'h10)^
({{8{data_in[224][6]}} & 8'h20)^
({{8{data_in[224][7]}} & 8'h40)^
({{8{data_in[225][0]}} & 8'he5)^
({{8{data_in[225][1]}} & 8'he1)^
({{8{data_in[225][2]}} & 8'he9)^
({{8{data_in[225][3]}} & 8'hf9)^
({{8{data_in[225][4]}} & 8'hd9)^
({{8{data_in[225][5]}} & 8'h99)^
({{8{data_in[225][6]}} & 8'h19)^
({{8{data_in[225][7]}} & 8'h32)^
({{8{data_in[226][0]}} & 8'ha1)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[226][1]}} & 8'h69)^
({{8{data_in[226][2]}} & 8'hd2)^
({{8{data_in[226][3]}} & 8'h8f)^
({{8{data_in[226][4]}} & 8'h35)^
({{8{data_in[226][5]}} & 8'h6a)^
({{8{data_in[226][6]}} & 8'hd4)^
({{8{data_in[226][7]}} & 8'h83)^
({{8{data_in[227][0]}} & 8'ha2)^
({{8{data_in[227][1]}} & 8'h6f)^
({{8{data_in[227][2]}} & 8'hde)^
({{8{data_in[227][3]}} & 8'h97)^
({{8{data_in[227][4]}} & 8'h5)^
({{8{data_in[227][5]}} & 8'ha)^
({{8{data_in[227][6]}} & 8'h14)^
({{8{data_in[227][7]}} & 8'h28)^
({{8{data_in[228][0]}} & 8'h2c)^
({{8{data_in[228][1]}} & 8'h58)^
({{8{data_in[228][2]}} & 8'hb0)^
({{8{data_in[228][3]}} & 8'h4b)^
({{8{data_in[228][4]}} & 8'h96)^
({{8{data_in[228][5]}} & 8'h7)^
({{8{data_in[228][6]}} & 8'he)^
({{8{data_in[228][7]}} & 8'h1c)^
({{8{data_in[229][0]}} & 8'h67)^
({{8{data_in[229][1]}} & 8'hce)^
({{8{data_in[229][2]}} & 8'hb7)^
({{8{data_in[229][3]}} & 8'h45)^
({{8{data_in[229][4]}} & 8'h8a)^
({{8{data_in[229][5]}} & 8'h3f)^
({{8{data_in[229][6]}} & 8'h7e)^
({{8{data_in[229][7]}} & 8'hfc)^
({{8{data_in[230][0]}} & 8'he9)^
({{8{data_in[230][1]}} & 8'hf9)^
({{8{data_in[230][2]}} & 8'hd9)^
({{8{data_in[230][3]}} & 8'h99)^
({{8{data_in[230][4]}} & 8'h19)^
({{8{data_in[230][5]}} & 8'h32)^
({{8{data_in[230][6]}} & 8'h64)^
({{8{data_in[230][7]}} & 8'hc8)^
({{8{data_in[231][0]}} & 8'h23)^
({{8{data_in[231][1]}} & 8'h46)^
({{8{data_in[231][2]}} & 8'h8c)^
({{8{data_in[231][3]}} & 8'h33)^
({{8{data_in[231][4]}} & 8'h66)^
({{8{data_in[231][5]}} & 8'hcc)^
({{8{data_in[231][6]}} & 8'hb3)^
({{8{data_in[231][7]}} & 8'h4d)^
({{8{data_in[232][0]}} & 8'h50)^
({{8{data_in[232][1]}} & 8'ha0)^
({{8{data_in[232][2]}} & 8'h6b)^
({{8{data_in[232][3]}} & 8'hd6)^
({{8{data_in[232][4]}} & 8'h87)^
({{8{data_in[232][5]}} & 8'h25)^
({{8{data_in[232][6]}} & 8'h4a)^
({{8{data_in[232][7]}} & 8'h94)^
({{8{data_in[233][0]}} & 8'h26)^
({{8{data_in[233][1]}} & 8'h4c)^
({{8{data_in[233][2]}} & 8'h98)^
({{8{data_in[233][3]}} & 8'h1b)^
({{8{data_in[233][4]}} & 8'h36)^
({{8{data_in[233][5]}} & 8'h6c)^
({{8{data_in[233][6]}} & 8'hd8)^
({{8{data_in[233][7]}} & 8'h9b)^
({{8{data_in[234][0]}} & 8'h51)^
({{8{data_in[234][1]}} & 8'ha2)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[234][2]}} & 8'h6f)^
({{8{data_in[234][3]}} & 8'hde)^
({{8{data_in[234][4]}} & 8'h97)^
({{8{data_in[234][5]}} & 8'h5)^
({{8{data_in[234][6]}} & 8'ha)^
({{8{data_in[234][7]}} & 8'h14)^
({{8{data_in[235][0]}} & 8'hd8)^
({{8{data_in[235][1]}} & 8'h9b)^
({{8{data_in[235][2]}} & 8'h1d)^
({{8{data_in[235][3]}} & 8'h3a)^
({{8{data_in[235][4]}} & 8'h74)^
({{8{data_in[235][5]}} & 8'he8)^
({{8{data_in[235][6]}} & 8'hfb)^
({{8{data_in[235][7]}} & 8'hdd)^
({{8{data_in[236][0]}} & 8'hc8)^
({{8{data_in[236][1]}} & 8'hbb)^
({{8{data_in[236][2]}} & 8'h5d)^
({{8{data_in[236][3]}} & 8'hba)^
({{8{data_in[236][4]}} & 8'h5f)^
({{8{data_in[236][5]}} & 8'hbe)^
({{8{data_in[236][6]}} & 8'h57)^
({{8{data_in[236][7]}} & 8'hae)^
({{8{data_in[237][0]}} & 8'h61)^
({{8{data_in[237][1]}} & 8'hc2)^
({{8{data_in[237][2]}} & 8'haf)^
({{8{data_in[237][3]}} & 8'h75)^
({{8{data_in[237][4]}} & 8'hea)^
({{8{data_in[237][5]}} & 8'hff)^
({{8{data_in[237][6]}} & 8'hd5)^
({{8{data_in[237][7]}} & 8'h81)^
({{8{data_in[238][0]}} & 8'hb3)^
({{8{data_in[238][1]}} & 8'h4d)^
({{8{data_in[238][2]}} & 8'h9a)^
({{8{data_in[238][3]}} & 8'h1f)^
({{8{data_in[238][4]}} & 8'h3e)^
({{8{data_in[238][5]}} & 8'h7c)^
({{8{data_in[238][6]}} & 8'hf8)^
({{8{data_in[238][7]}} & 8'hdb)^
({{8{data_in[239][0]}} & 8'h5a)^
({{8{data_in[239][1]}} & 8'hb4)^
({{8{data_in[239][2]}} & 8'h43)^
({{8{data_in[239][3]}} & 8'h86)^
({{8{data_in[239][4]}} & 8'h27)^
({{8{data_in[239][5]}} & 8'h4e)^
({{8{data_in[239][6]}} & 8'h9c)^
({{8{data_in[239][7]}} & 8'h13)^
({{8{data_in[240][0]}} & 8'h5e)^
({{8{data_in[240][1]}} & 8'hbc)^
({{8{data_in[240][2]}} & 8'h53)^
({{8{data_in[240][3]}} & 8'ha6)^
({{8{data_in[240][4]}} & 8'h67)^
({{8{data_in[240][5]}} & 8'hce)^
({{8{data_in[240][6]}} & 8'hb7)^
({{8{data_in[240][7]}} & 8'h45)^
({{8{data_in[241][0]}} & 8'h41)^
({{8{data_in[241][1]}} & 8'h82)^
({{8{data_in[241][2]}} & 8'h2f)^
({{8{data_in[241][3]}} & 8'h5e)^
({{8{data_in[241][4]}} & 8'hbc)^
({{8{data_in[241][5]}} & 8'h53)^
({{8{data_in[241][6]}} & 8'ha6)^
({{8{data_in[241][7]}} & 8'h67);

data_out[243] = ({8{data_in[0][0]}} & 8'h3b)^
    ({8{data_in[0][1]}} & 8'h76)^
    ({8{data_in[0][2]}} & 8'hec)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[0][3]}} & 8'hf3)^
({{8{data_in[0][4]}} & 8'hcd)^
({{8{data_in[0][5]}} & 8'hb1)^
({{8{data_in[0][6]}} & 8'h49)^
({{8{data_in[0][7]}} & 8'h92)^
({{8{data_in[1][0]}} & 8'h29)^
({{8{data_in[1][1]}} & 8'h52)^
({{8{data_in[1][2]}} & 8'ha4)^
({{8{data_in[1][3]}} & 8'h63)^
({{8{data_in[1][4]}} & 8'hc6)^
({{8{data_in[1][5]}} & 8'ha7)^
({{8{data_in[1][6]}} & 8'h65)^
({{8{data_in[1][7]}} & 8'hca)^
({{8{data_in[2][0]}} & 8'h11)^
({{8{data_in[2][1]}} & 8'h22)^
({{8{data_in[2][2]}} & 8'h44)^
({{8{data_in[2][3]}} & 8'h88)^
({{8{data_in[2][4]}} & 8'h3b)^
({{8{data_in[2][5]}} & 8'h76)^
({{8{data_in[2][6]}} & 8'hec)^
({{8{data_in[2][7]}} & 8'hf3)^
({{8{data_in[3][0]}} & 8'h56)^
({{8{data_in[3][1]}} & 8'hac)^
({{8{data_in[3][2]}} & 8'h73)^
({{8{data_in[3][3]}} & 8'he6)^
({{8{data_in[3][4]}} & 8'he7)^
({{8{data_in[3][5]}} & 8'he5)^
({{8{data_in[3][6]}} & 8'he1)^
({{8{data_in[3][7]}} & 8'he9)^
({{8{data_in[4][0]}} & 8'h44)^
({{8{data_in[4][1]}} & 8'h88)^
({{8{data_in[4][2]}} & 8'h3b)^
({{8{data_in[4][3]}} & 8'h76)^
({{8{data_in[4][4]}} & 8'hec)^
({{8{data_in[4][5]}} & 8'hf3)^
({{8{data_in[4][6]}} & 8'hcd)^
({{8{data_in[4][7]}} & 8'hb1)^
({{8{data_in[5][0]}} & 8'hec)^
({{8{data_in[5][1]}} & 8'hf3)^
({{8{data_in[5][2]}} & 8'hcd)^
({{8{data_in[5][3]}} & 8'hb1)^
({{8{data_in[5][4]}} & 8'h49)^
({{8{data_in[5][5]}} & 8'h92)^
({{8{data_in[5][6]}} & 8'hf)^
({{8{data_in[5][7]}} & 8'h1e)^
({{8{data_in[6][0]}} & 8'hea)^
({{8{data_in[6][1]}} & 8'hff)^
({{8{data_in[6][2]}} & 8'hd5)^
({{8{data_in[6][3]}} & 8'h81)^
({{8{data_in[6][4]}} & 8'h29)^
({{8{data_in[6][5]}} & 8'h52)^
({{8{data_in[6][6]}} & 8'ha4)^
({{8{data_in[6][7]}} & 8'h63)^
({{8{data_in[7][0]}} & 8'h3a)^
({{8{data_in[7][1]}} & 8'h74)^
({{8{data_in[7][2]}} & 8'he8)^
({{8{data_in[7][3]}} & 8'hfb)^
({{8{data_in[7][4]}} & 8'hdd)^
({{8{data_in[7][5]}} & 8'h91)^
({{8{data_in[7][6]}} & 8'h9)^
({{8{data_in[7][7]}} & 8'h12)^
({{8{data_in[8][0]}} & 8'h3)^
({{8{data_in[8][1]}} & 8'h6)^
({{8{data_in[8][2]}} & 8'hc)^
({{8{data_in[8][3]}} & 8'h18)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[8][4]}} & 8'h30)^
({{8{data_in[8][5]}} & 8'h60)^
({{8{data_in[8][6]}} & 8'hc0)^
({{8{data_in[8][7]}} & 8'hab)^
({{8{data_in[9][0]}} & 8'h72)^
({{8{data_in[9][1]}} & 8'he4)^
({{8{data_in[9][2]}} & 8'he3)^
({{8{data_in[9][3]}} & 8'hed)^
({{8{data_in[9][4]}} & 8'hf1)^
({{8{data_in[9][5]}} & 8'hc9)^
({{8{data_in[9][6]}} & 8'hb9)^
({{8{data_in[9][7]}} & 8'h59)^
({{8{data_in[10][0]}} & 8'h11)^
({{8{data_in[10][1]}} & 8'h22)^
({{8{data_in[10][2]}} & 8'h44)^
({{8{data_in[10][3]}} & 8'h88)^
({{8{data_in[10][4]}} & 8'h3b)^
({{8{data_in[10][5]}} & 8'h76)^
({{8{data_in[10][6]}} & 8'hec)^
({{8{data_in[10][7]}} & 8'hf3)^
({{8{data_in[11][0]}} & 8'h4c)^
({{8{data_in[11][1]}} & 8'h98)^
({{8{data_in[11][2]}} & 8'h1b)^
({{8{data_in[11][3]}} & 8'h36)^
({{8{data_in[11][4]}} & 8'h6c)^
({{8{data_in[11][5]}} & 8'hd8)^
({{8{data_in[11][6]}} & 8'h9b)^
({{8{data_in[11][7]}} & 8'h1d)^
({{8{data_in[12][0]}} & 8'h7c)^
({{8{data_in[12][1]}} & 8'hf8)^
({{8{data_in[12][2]}} & 8'hdb)^
({{8{data_in[12][3]}} & 8'h9d)^
({{8{data_in[12][4]}} & 8'h11)^
({{8{data_in[12][5]}} & 8'h22)^
({{8{data_in[12][6]}} & 8'h44)^
({{8{data_in[12][7]}} & 8'h88)^
({{8{data_in[13][0]}} & 8'hcb)^
({{8{data_in[13][1]}} & 8'hbd)^
({{8{data_in[13][2]}} & 8'h51)^
({{8{data_in[13][3]}} & 8'ha2)^
({{8{data_in[13][4]}} & 8'h6f)^
({{8{data_in[13][5]}} & 8'hde)^
({{8{data_in[13][6]}} & 8'h97)^
({{8{data_in[13][7]}} & 8'h5)^
({{8{data_in[14][0]}} & 8'hc9)^
({{8{data_in[14][1]}} & 8'hb9)^
({{8{data_in[14][2]}} & 8'h59)^
({{8{data_in[14][3]}} & 8'hb2)^
({{8{data_in[14][4]}} & 8'h4f)^
({{8{data_in[14][5]}} & 8'h9e)^
({{8{data_in[14][6]}} & 8'h17)^
({{8{data_in[14][7]}} & 8'h2e)^
({{8{data_in[15][0]}} & 8'h8f)^
({{8{data_in[15][1]}} & 8'h35)^
({{8{data_in[15][2]}} & 8'h6a)^
({{8{data_in[15][3]}} & 8'hd4)^
({{8{data_in[15][4]}} & 8'h83)^
({{8{data_in[15][5]}} & 8'h2d)^
({{8{data_in[15][6]}} & 8'h5a)^
({{8{data_in[15][7]}} & 8'hb4)^
({{8{data_in[16][0]}} & 8'haa)^
({{8{data_in[16][1]}} & 8'h7f)^
({{8{data_in[16][2]}} & 8'hfe)^
({{8{data_in[16][3]}} & 8'hd7)^
({{8{data_in[16][4]}} & 8'h85)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[16][5]}} & 8'h21)^
({{8{data_in[16][6]}} & 8'h42)^
({{8{data_in[16][7]}} & 8'h84)^
({{8{data_in[17][0]}} & 8'h14)^
({{8{data_in[17][1]}} & 8'h28)^
({{8{data_in[17][2]}} & 8'h50)^
({{8{data_in[17][3]}} & 8'ha0)^
({{8{data_in[17][4]}} & 8'h6b)^
({{8{data_in[17][5]}} & 8'hd6)^
({{8{data_in[17][6]}} & 8'h87)^
({{8{data_in[17][7]}} & 8'h25)^
({{8{data_in[18][0]}} & 8'hac)^
({{8{data_in[18][1]}} & 8'h73)^
({{8{data_in[18][2]}} & 8'he6)^
({{8{data_in[18][3]}} & 8'he7)^
({{8{data_in[18][4]}} & 8'he5)^
({{8{data_in[18][5]}} & 8'he1)^
({{8{data_in[18][6]}} & 8'he9)^
({{8{data_in[18][7]}} & 8'hf9)^
({{8{data_in[19][0]}} & 8'hca)^
({{8{data_in[19][1]}} & 8'hbf)^
({{8{data_in[19][2]}} & 8'h55)^
({{8{data_in[19][3]}} & 8'haa)^
({{8{data_in[19][4]}} & 8'h7f)^
({{8{data_in[19][5]}} & 8'hfe)^
({{8{data_in[19][6]}} & 8'hd7)^
({{8{data_in[19][7]}} & 8'h85)^
({{8{data_in[20][0]}} & 8'h7a)^
({{8{data_in[20][1]}} & 8'hf4)^
({{8{data_in[20][2]}} & 8'hc3)^
({{8{data_in[20][3]}} & 8'had)^
({{8{data_in[20][4]}} & 8'h71)^
({{8{data_in[20][5]}} & 8'he2)^
({{8{data_in[20][6]}} & 8'hef)^
({{8{data_in[20][7]}} & 8'hf5)^
({{8{data_in[21][0]}} & 8'h66)^
({{8{data_in[21][1]}} & 8'hcc)^
({{8{data_in[21][2]}} & 8'hb3)^
({{8{data_in[21][3]}} & 8'h4d)^
({{8{data_in[21][4]}} & 8'�9a)^
({{8{data_in[21][5]}} & 8'h1f)^
({{8{data_in[21][6]}} & 8'h3e)^
({{8{data_in[21][7]}} & 8'h7c)^
({{8{data_in[22][0]}} & 8'hdb)^
({{8{data_in[22][1]}} & 8'h9d)^
({{8{data_in[22][2]}} & 8'h11)^
({{8{data_in[22][3]}} & 8'h22)^
({{8{data_in[22][4]}} & 8'h44)^
({{8{data_in[22][5]}} & 8'h88)^
({{8{data_in[22][6]}} & 8'h3b)^
({{8{data_in[22][7]}} & 8'h76)^
({{8{data_in[23][0]}} & 8'ha2)^
({{8{data_in[23][1]}} & 8'h6f)^
({{8{data_in[23][2]}} & 8'hde)^
({{8{data_in[23][3]}} & 8'h97)^
({{8{data_in[23][4]}} & 8'h5)^
({{8{data_in[23][5]}} & 8'ha)^
({{8{data_in[23][6]}} & 8'h14)^
({{8{data_in[23][7]}} & 8'h28)^
({{8{data_in[24][0]}} & 8'h1d)^
({{8{data_in[24][1]}} & 8'h3a)^
({{8{data_in[24][2]}} & 8'h74)^
({{8{data_in[24][3]}} & 8'he8)^
({{8{data_in[24][4]}} & 8'hfb)^
({{8{data_in[24][5]}} & 8'hdd)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[24][6]}} & 8'h91)^
({{8{data_in[24][7]}} & 8'h9)^
({{8{data_in[25][0]}} & 8'h96)^
({{8{data_in[25][1]}} & 8'h7)^
({{8{data_in[25][2]}} & 8'he)^
({{8{data_in[25][3]}} & 8'h1c)^
({{8{data_in[25][4]}} & 8'h38)^
({{8{data_in[25][5]}} & 8'h70)^
({{8{data_in[25][6]}} & 8'he0)^
({{8{data_in[25][7]}} & 8'heb)^
({{8{data_in[26][0]}} & 8'h84)^
({{8{data_in[26][1]}} & 8'h23)^
({{8{data_in[26][2]}} & 8'h46)^
({{8{data_in[26][3]}} & 8'h8c)^
({{8{data_in[26][4]}} & 8'h33)^
({{8{data_in[26][5]}} & 8'h66)^
({{8{data_in[26][6]}} & 8'hcc)^
({{8{data_in[26][7]}} & 8'hb3)^
({{8{data_in[27][0]}} & 8'h4e)^
({{8{data_in[27][1]}} & 8'h9c)^
({{8{data_in[27][2]}} & 8'h13)^
({{8{data_in[27][3]}} & 8'h26)^
({{8{data_in[27][4]}} & 8'h4c)^
({{8{data_in[27][5]}} & 8'h98)^
({{8{data_in[27][6]}} & 8'h1b)^
({{8{data_in[27][7]}} & 8'h36)^
({{8{data_in[28][0]}} & 8'h78)^
({{8{data_in[28][1]}} & 8'hf0)^
({{8{data_in[28][2]}} & 8'hcb)^
({{8{data_in[28][3]}} & 8'hbd)^
({{8{data_in[28][4]}} & 8'h51)^
({{8{data_in[28][5]}} & 8'ha2)^
({{8{data_in[28][6]}} & 8'h6f)^
({{8{data_in[28][7]}} & 8'hde)^
({{8{data_in[29][0]}} & 8'h3a)^
({{8{data_in[29][1]}} & 8'h74)^
({{8{data_in[29][2]}} & 8'he8)^
({{8{data_in[29][3]}} & 8'hfb)^
({{8{data_in[29][4]}} & 8'hdd)^
({{8{data_in[29][5]}} & 8'h91)^
({{8{data_in[29][6]}} & 8'h9)^
({{8{data_in[29][7]}} & 8'h12)^
({{8{data_in[30][0]}} & 8'hd)^
({{8{data_in[30][1]}} & 8'h1a)^
({{8{data_in[30][2]}} & 8'h34)^
({{8{data_in[30][3]}} & 8'h68)^
({{8{data_in[30][4]}} & 8'hd0)^
({{8{data_in[30][5]}} & 8'h8b)^
({{8{data_in[30][6]}} & 8'h3d)^
({{8{data_in[30][7]}} & 8'h7a)^
({{8{data_in[31][0]}} & 8'h82)^
({{8{data_in[31][1]}} & 8'h2f)^
({{8{data_in[31][2]}} & 8'h5e)^
({{8{data_in[31][3]}} & 8'hbc)^
({{8{data_in[31][4]}} & 8'h53)^
({{8{data_in[31][5]}} & 8'ha6)^
({{8{data_in[31][6]}} & 8'h67)^
({{8{data_in[31][7]}} & 8'hce)^
({{8{data_in[32][0]}} & 8'hb8)^
({{8{data_in[32][1]}} & 8'h5b)^
({{8{data_in[32][2]}} & 8'hb6)^
({{8{data_in[32][3]}} & 8'h47)^
({{8{data_in[32][4]}} & 8'h8e)^
({{8{data_in[32][5]}} & 8'h37)^
({{8{data_in[32][6]}} & 8'h6e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[32][7]}} & 8'hdc)^
({{8{data_in[33][0]}} & 8'h36)^
({{8{data_in[33][1]}} & 8'h6c)^
({{8{data_in[33][2]}} & 8'hd8)^
({{8{data_in[33][3]}} & 8'h9b)^
({{8{data_in[33][4]}} & 8'h1d)^
({{8{data_in[33][5]}} & 8'h3a)^
({{8{data_in[33][6]}} & 8'h74)^
({{8{data_in[33][7]}} & 8'he8)^
({{8{data_in[34][0]}} & 8'h6b)^
({{8{data_in[34][1]}} & 8'hd6)^
({{8{data_in[34][2]}} & 8'h87)^
({{8{data_in[34][3]}} & 8'h25)^
({{8{data_in[34][4]}} & 8'h4a)^
({{8{data_in[34][5]}} & 8'h94)^
({{8{data_in[34][6]}} & 8'h3)^
({{8{data_in[34][7]}} & 8'h6)^
({{8{data_in[35][0]}} & 8'h1c)^
({{8{data_in[35][1]}} & 8'h38)^
({{8{data_in[35][2]}} & 8'h70)^
({{8{data_in[35][3]}} & 8'he0)^
({{8{data_in[35][4]}} & 8'heb)^
({{8{data_in[35][5]}} & 8'hfd)^
({{8{data_in[35][6]}} & 8'hd1)^
({{8{data_in[35][7]}} & 8'h89)^
({{8{data_in[36][0]}} & 8'h3a)^
({{8{data_in[36][1]}} & 8'h74)^
({{8{data_in[36][2]}} & 8'he8)^
({{8{data_in[36][3]}} & 8'hfb)^
({{8{data_in[36][4]}} & 8'hdd)^
({{8{data_in[36][5]}} & 8'h91)^
({{8{data_in[36][6]}} & 8'h9)^
({{8{data_in[36][7]}} & 8'h12)^
({{8{data_in[37][0]}} & 8'hed)^
({{8{data_in[37][1]}} & 8'hf1)^
({{8{data_in[37][2]}} & 8'hc9)^
({{8{data_in[37][3]}} & 8'hb9)^
({{8{data_in[37][4]}} & 8'h59)^
({{8{data_in[37][5]}} & 8'hb2)^
({{8{data_in[37][6]}} & 8'h4f)^
({{8{data_in[37][7]}} & 8'h9e)^
({{8{data_in[38][0]}} & 8'h13)^
({{8{data_in[38][1]}} & 8'h26)^
({{8{data_in[38][2]}} & 8'h4c)^
({{8{data_in[38][3]}} & 8'h98)^
({{8{data_in[38][4]}} & 8'h1b)^
({{8{data_in[38][5]}} & 8'h36)^
({{8{data_in[38][6]}} & 8'h6c)^
({{8{data_in[38][7]}} & 8'hd8)^
({{8{data_in[39][0]}} & 8'haa)^
({{8{data_in[39][1]}} & 8'h7f)^
({{8{data_in[39][2]}} & 8'hfe)^
({{8{data_in[39][3]}} & 8'hd7)^
({{8{data_in[39][4]}} & 8'h85)^
({{8{data_in[39][5]}} & 8'h21)^
({{8{data_in[39][6]}} & 8'h42)^
({{8{data_in[39][7]}} & 8'h84)^
({{8{data_in[40][0]}} & 8'h46)^
({{8{data_in[40][1]}} & 8'h8c)^
({{8{data_in[40][2]}} & 8'h33)^
({{8{data_in[40][3]}} & 8'h66)^
({{8{data_in[40][4]}} & 8'hcc)^
({{8{data_in[40][5]}} & 8'hb3)^
({{8{data_in[40][6]}} & 8'h4d)^
({{8{data_in[40][7]}} & 8'h9a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[41][0]}} & 8'h1f)^
({{8{data_in[41][1]}} & 8'h3e)^
({{8{data_in[41][2]}} & 8'h7c)^
({{8{data_in[41][3]}} & 8'hf8)^
({{8{data_in[41][4]}} & 8'hdb)^
({{8{data_in[41][5]}} & 8'h9d)^
({{8{data_in[41][6]}} & 8'h11)^
({{8{data_in[41][7]}} & 8'h22)^
({{8{data_in[42][0]}} & 8'h32)^
({{8{data_in[42][1]}} & 8'h64)^
({{8{data_in[42][2]}} & 8'hc8)^
({{8{data_in[42][3]}} & 8'hbb)^
({{8{data_in[42][4]}} & 8'h5d)^
({{8{data_in[42][5]}} & 8'hba)^
({{8{data_in[42][6]}} & 8'h5f)^
({{8{data_in[42][7]}} & 8'hbe)^
({{8{data_in[43][0]}} & 8'h92)^
({{8{data_in[43][1]}} & 8'hf)^
({{8{data_in[43][2]}} & 8'h1e)^
({{8{data_in[43][3]}} & 8'h3c)^
({{8{data_in[43][4]}} & 8'h78)^
({{8{data_in[43][5]}} & 8'hf0)^
({{8{data_in[43][6]}} & 8'hcb)^
({{8{data_in[43][7]}} & 8'hbd)^
({{8{data_in[44][0]}} & 8'hd3)^
({{8{data_in[44][1]}} & 8'h8d)^
({{8{data_in[44][2]}} & 8'h31)^
({{8{data_in[44][3]}} & 8'h62)^
({{8{data_in[44][4]}} & 8'hc4)^
({{8{data_in[44][5]}} & 8'ha3)^
({{8{data_in[44][6]}} & 8'h6d)^
({{8{data_in[44][7]}} & 8'hda)^
({{8{data_in[45][0]}} & 8'h6f)^
({{8{data_in[45][1]}} & 8'hde)^
({{8{data_in[45][2]}} & 8'h97)^
({{8{data_in[45][3]}} & 8'h5)^
({{8{data_in[45][4]}} & 8'ha)^
({{8{data_in[45][5]}} & 8'h14)^
({{8{data_in[45][6]}} & 8'h28)^
({{8{data_in[45][7]}} & 8'h50)^
({{8{data_in[46][0]}} & 8'h1f)^
({{8{data_in[46][1]}} & 8'h3e)^
({{8{data_in[46][2]}} & 8'h7c)^
({{8{data_in[46][3]}} & 8'hf8)^
({{8{data_in[46][4]}} & 8'hdb)^
({{8{data_in[46][5]}} & 8'h9d)^
({{8{data_in[46][6]}} & 8'h11)^
({{8{data_in[46][7]}} & 8'h22)^
({{8{data_in[47][0]}} & 8'h31)^
({{8{data_in[47][1]}} & 8'h62)^
({{8{data_in[47][2]}} & 8'hc4)^
({{8{data_in[47][3]}} & 8'ha3)^
({{8{data_in[47][4]}} & 8'h6d)^
({{8{data_in[47][5]}} & 8'hda)^
({{8{data_in[47][6]}} & 8'h9f)^
({{8{data_in[47][7]}} & 8'h15)^
({{8{data_in[48][0]}} & 8'h1d)^
({{8{data_in[48][1]}} & 8'h3a)^
({{8{data_in[48][2]}} & 8'h74)^
({{8{data_in[48][3]}} & 8'he8)^
({{8{data_in[48][4]}} & 8'hfb)^
({{8{data_in[48][5]}} & 8'hdd)^
({{8{data_in[48][6]}} & 8'h91)^
({{8{data_in[48][7]}} & 8'h9)^
({{8{data_in[49][0]}} & 8'h11)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[49][1]}} & 8'h22)^
({{8{data_in[49][2]}} & 8'h44)^
({{8{data_in[49][3]}} & 8'h88)^
({{8{data_in[49][4]}} & 8'h3b)^
({{8{data_in[49][5]}} & 8'h76)^
({{8{data_in[49][6]}} & 8'hec)^
({{8{data_in[49][7]}} & 8'hf3)^
({{8{data_in[50][0]}} & 8'h1b)^
({{8{data_in[50][1]}} & 8'h36)^
({{8{data_in[50][2]}} & 8'h6c)^
({{8{data_in[50][3]}} & 8'hd8)^
({{8{data_in[50][4]}} & 8'h9b)^
({{8{data_in[50][5]}} & 8'h1d)^
({{8{data_in[50][6]}} & 8'h3a)^
({{8{data_in[50][7]}} & 8'h74)^
({{8{data_in[51][0]}} & 8'hb)^
({{8{data_in[51][1]}} & 8'h16)^
({{8{data_in[51][2]}} & 8'h2c)^
({{8{data_in[51][3]}} & 8'h58)^
({{8{data_in[51][4]}} & 8'hb0)^
({{8{data_in[51][5]}} & 8'h4b)^
({{8{data_in[51][6]}} & 8'h96)^
({{8{data_in[51][7]}} & 8'h7)^
({{8{data_in[52][0]}} & 8'hed)^
({{8{data_in[52][1]}} & 8'hf1)^
({{8{data_in[52][2]}} & 8'hc9)^
({{8{data_in[52][3]}} & 8'hb9)^
({{8{data_in[52][4]}} & 8'h59)^
({{8{data_in[52][5]}} & 8'hb2)^
({{8{data_in[52][6]}} & 8'h4f)^
({{8{data_in[52][7]}} & 8'h9e)^
({{8{data_in[53][0]}} & 8'h4)^
({{8{data_in[53][1]}} & 8'h8)^
({{8{data_in[53][2]}} & 8'h10)^
({{8{data_in[53][3]}} & 8'h20)^
({{8{data_in[53][4]}} & 8'h40)^
({{8{data_in[53][5]}} & 8'h80)^
({{8{data_in[53][6]}} & 8'h2b)^
({{8{data_in[53][7]}} & 8'h56)^
({{8{data_in[54][0]}} & 8'h63)^
({{8{data_in[54][1]}} & 8'hc6)^
({{8{data_in[54][2]}} & 8'ha7)^
({{8{data_in[54][3]}} & 8'h65)^
({{8{data_in[54][4]}} & 8'hca)^
({{8{data_in[54][5]}} & 8'hbf)^
({{8{data_in[54][6]}} & 8'h55)^
({{8{data_in[54][7]}} & 8'haa)^
({{8{data_in[55][0]}} & 8'hca)^
({{8{data_in[55][1]}} & 8'hbf)^
({{8{data_in[55][2]}} & 8'h55)^
({{8{data_in[55][3]}} & 8'haa)^
({{8{data_in[55][4]}} & 8'h7f)^
({{8{data_in[55][5]}} & 8'hfe)^
({{8{data_in[55][6]}} & 8'hd7)^
({{8{data_in[55][7]}} & 8'h85)^
({{8{data_in[56][0]}} & 8'h9e)^
({{8{data_in[56][1]}} & 8'h17)^
({{8{data_in[56][2]}} & 8'h2e)^
({{8{data_in[56][3]}} & 8'h5c)^
({{8{data_in[56][4]}} & 8'hb8)^
({{8{data_in[56][5]}} & 8'h5b)^
({{8{data_in[56][6]}} & 8'hb6)^
({{8{data_in[56][7]}} & 8'h47)^
({{8{data_in[57][0]}} & 8'h2a)^
({{8{data_in[57][1]}} & 8'h54)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[57][2]}} & 8'ha8)^
({{8{data_in[57][3]}} & 8'h7b)^
({{8{data_in[57][4]}} & 8'hf6)^
({{8{data_in[57][5]}} & 8'hc7)^
({{8{data_in[57][6]}} & 8'ha5)^
({{8{data_in[57][7]}} & 8'h61)^
({{8{data_in[58][0]}} & 8'hd1)^
({{8{data_in[58][1]}} & 8'h89)^
({{8{data_in[58][2]}} & 8'h39)^
({{8{data_in[58][3]}} & 8'h72)^
({{8{data_in[58][4]}} & 8'he4)^
({{8{data_in[58][5]}} & 8'he3)^
({{8{data_in[58][6]}} & 8'hed)^
({{8{data_in[58][7]}} & 8'hf1)^
({{8{data_in[59][0]}} & 8'h4f)^
({{8{data_in[59][1]}} & 8'h9e)^
({{8{data_in[59][2]}} & 8'h17)^
({{8{data_in[59][3]}} & 8'h2e)^
({{8{data_in[59][4]}} & 8'h5c)^
({{8{data_in[59][5]}} & 8'hb8)^
({{8{data_in[59][6]}} & 8'h5b)^
({{8{data_in[59][7]}} & 8'hb6)^
({{8{data_in[60][0]}} & 8'hd3)^
({{8{data_in[60][1]}} & 8'h8d)^
({{8{data_in[60][2]}} & 8'h31)^
({{8{data_in[60][3]}} & 8'h62)^
({{8{data_in[60][4]}} & 8'hc4)^
({{8{data_in[60][5]}} & 8'ha3)^
({{8{data_in[60][6]}} & 8'h6d)^
({{8{data_in[60][7]}} & 8'hda)^
({{8{data_in[61][0]}} & 8'hdd)^
({{8{data_in[61][1]}} & 8'h91)^
({{8{data_in[61][2]}} & 8'h9)^
({{8{data_in[61][3]}} & 8'h12)^
({{8{data_in[61][4]}} & 8'h24)^
({{8{data_in[61][5]}} & 8'h48)^
({{8{data_in[61][6]}} & 8'h90)^
({{8{data_in[61][7]}} & 8'hb)^
({{8{data_in[62][0]}} & 8'h49)^
({{8{data_in[62][1]}} & 8'h92)^
({{8{data_in[62][2]}} & 8'hf)^
({{8{data_in[62][3]}} & 8'h1e)^
({{8{data_in[62][4]}} & 8'h3c)^
({{8{data_in[62][5]}} & 8'h78)^
({{8{data_in[62][6]}} & 8'hf0)^
({{8{data_in[62][7]}} & 8'hcb)^
({{8{data_in[63][0]}} & 8'hce)^
({{8{data_in[63][1]}} & 8'hb7)^
({{8{data_in[63][2]}} & 8'h45)^
({{8{data_in[63][3]}} & 8'h8a)^
({{8{data_in[63][4]}} & 8'h3f)^
({{8{data_in[63][5]}} & 8'h7e)^
({{8{data_in[63][6]}} & 8'hfc)^
({{8{data_in[63][7]}} & 8'hd3)^
({{8{data_in[64][0]}} & 8'h77)^
({{8{data_in[64][1]}} & 8'hee)^
({{8{data_in[64][2]}} & 8'hf7)^
({{8{data_in[64][3]}} & 8'hc5)^
({{8{data_in[64][4]}} & 8'ha1)^
({{8{data_in[64][5]}} & 8'h69)^
({{8{data_in[64][6]}} & 8'hd2)^
({{8{data_in[64][7]}} & 8'h8f)^
({{8{data_in[65][0]}} & 8'he2)^
({{8{data_in[65][1]}} & 8'hef)^
({{8{data_in[65][2]}} & 8'hf5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[65][3]}} & 8'hc1)^
({{8{data_in[65][4]}} & 8'ha9)^
({{8{data_in[65][5]}} & 8'h79)^
({{8{data_in[65][6]}} & 8'hf2)^
({{8{data_in[65][7]}} & 8'hcf)^
({{8{data_in[66][0]}} & 8'hff)^
({{8{data_in[66][1]}} & 8'hd5)^
({{8{data_in[66][2]}} & 8'h81)^
({{8{data_in[66][3]}} & 8'h29)^
({{8{data_in[66][4]}} & 8'h52)^
({{8{data_in[66][5]}} & 8'ha4)^
({{8{data_in[66][6]}} & 8'h63)^
({{8{data_in[66][7]}} & 8'hc6)^
({{8{data_in[67][0]}} & 8'h22)^
({{8{data_in[67][1]}} & 8'h44)^
({{8{data_in[67][2]}} & 8'h88)^
({{8{data_in[67][3]}} & 8'h3b)^
({{8{data_in[67][4]}} & 8'h76)^
({{8{data_in[67][5]}} & 8'hec)^
({{8{data_in[67][6]}} & 8'hf3)^
({{8{data_in[67][7]}} & 8'hcd)^
({{8{data_in[68][0]}} & 8'h5b)^
({{8{data_in[68][1]}} & 8'hb6)^
({{8{data_in[68][2]}} & 8'h47)^
({{8{data_in[68][3]}} & 8'h8e)^
({{8{data_in[68][4]}} & 8'h37)^
({{8{data_in[68][5]}} & 8'h6e)^
({{8{data_in[68][6]}} & 8'hdc)^
({{8{data_in[68][7]}} & 8'h93)^
({{8{data_in[69][0]}} & 8'hbb)^
({{8{data_in[69][1]}} & 8'h5d)^
({{8{data_in[69][2]}} & 8'hba)^
({{8{data_in[69][3]}} & 8'h5f)^
({{8{data_in[69][4]}} & 8'hbe)^
({{8{data_in[69][5]}} & 8'h57)^
({{8{data_in[69][6]}} & 8'hae)^
({{8{data_in[69][7]}} & 8'h77)^
({{8{data_in[70][0]}} & 8'hd9)^
({{8{data_in[70][1]}} & 8'h99)^
({{8{data_in[70][2]}} & 8'h19)^
({{8{data_in[70][3]}} & 8'h32)^
({{8{data_in[70][4]}} & 8'h64)^
({{8{data_in[70][5]}} & 8'hc8)^
({{8{data_in[70][6]}} & 8'hbb)^
({{8{data_in[70][7]}} & 8'h5d)^
({{8{data_in[71][0]}} & 8'h58)^
({{8{data_in[71][1]}} & 8'hb0)^
({{8{data_in[71][2]}} & 8'h4b)^
({{8{data_in[71][3]}} & 8'h96)^
({{8{data_in[71][4]}} & 8'h7)^
({{8{data_in[71][5]}} & 8'he)^
({{8{data_in[71][6]}} & 8'h1c)^
({{8{data_in[71][7]}} & 8'h38)^
({{8{data_in[72][0]}} & 8'h6b)^
({{8{data_in[72][1]}} & 8'hd6)^
({{8{data_in[72][2]}} & 8'h87)^
({{8{data_in[72][3]}} & 8'h25)^
({{8{data_in[72][4]}} & 8'h4a)^
({{8{data_in[72][5]}} & 8'h94)^
({{8{data_in[72][6]}} & 8'h3)^
({{8{data_in[72][7]}} & 8'h6)^
({{8{data_in[73][0]}} & 8'hd5)^
({{8{data_in[73][1]}} & 8'h81)^
({{8{data_in[73][2]}} & 8'h29)^
({{8{data_in[73][3]}} & 8'h52)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[73][4]}} & 8'ha4)^
({{8{data_in[73][5]}} & 8'h63)^
({{8{data_in[73][6]}} & 8'hc6)^
({{8{data_in[73][7]}} & 8'ha7)^
({{8{data_in[74][0]}} & 8'h66)^
({{8{data_in[74][1]}} & 8'hcc)^
({{8{data_in[74][2]}} & 8'hb3)^
({{8{data_in[74][3]}} & 8'h4d)^
({{8{data_in[74][4]}} & 8'h9a)^
({{8{data_in[74][5]}} & 8'h1f)^
({{8{data_in[74][6]}} & 8'h3e)^
({{8{data_in[74][7]}} & 8'h7c)^
({{8{data_in[75][0]}} & 8'h8d)^
({{8{data_in[75][1]}} & 8'h31)^
({{8{data_in[75][2]}} & 8'h62)^
({{8{data_in[75][3]}} & 8'hc4)^
({{8{data_in[75][4]}} & 8'ha3)^
({{8{data_in[75][5]}} & 8'h6d)^
({{8{data_in[75][6]}} & 8'hda)^
({{8{data_in[75][7]}} & 8'h9f)^
({{8{data_in[76][0]}} & 8'ha9)^
({{8{data_in[76][1]}} & 8'h79)^
({{8{data_in[76][2]}} & 8'hf2)^
({{8{data_in[76][3]}} & 8'hcf)^
({{8{data_in[76][4]}} & 8'hb5)^
({{8{data_in[76][5]}} & 8'h41)^
({{8{data_in[76][6]}} & 8'h82)^
({{8{data_in[76][7]}} & 8'h2f)^
({{8{data_in[77][0]}} & 8'h62)^
({{8{data_in[77][1]}} & 8'hc4)^
({{8{data_in[77][2]}} & 8'ha3)^
({{8{data_in[77][3]}} & 8'h6d)^
({{8{data_in[77][4]}} & 8'hda)^
({{8{data_in[77][5]}} & 8'h9f)^
({{8{data_in[77][6]}} & 8'h15)^
({{8{data_in[77][7]}} & 8'h2a)^
({{8{data_in[78][0]}} & 8'h41)^
({{8{data_in[78][1]}} & 8'h82)^
({{8{data_in[78][2]}} & 8'h2f)^
({{8{data_in[78][3]}} & 8'h5e)^
({{8{data_in[78][4]}} & 8'hbc)^
({{8{data_in[78][5]}} & 8'h53)^
({{8{data_in[78][6]}} & 8'ha6)^
({{8{data_in[78][7]}} & 8'h67)^
({{8{data_in[79][0]}} & 8'h8)^
({{8{data_in[79][1]}} & 8'h10)^
({{8{data_in[79][2]}} & 8'h20)^
({{8{data_in[79][3]}} & 8'h40)^
({{8{data_in[79][4]}} & 8'h80)^
({{8{data_in[79][5]}} & 8'h2b)^
({{8{data_in[79][6]}} & 8'h56)^
({{8{data_in[79][7]}} & 8'hac)^
({{8{data_in[80][0]}} & 8'h24)^
({{8{data_in[80][1]}} & 8'h48)^
({{8{data_in[80][2]}} & 8'h90)^
({{8{data_in[80][3]}} & 8'hb)^
({{8{data_in[80][4]}} & 8'h16)^
({{8{data_in[80][5]}} & 8'h2c)^
({{8{data_in[80][6]}} & 8'h58)^
({{8{data_in[80][7]}} & 8'hb0)^
({{8{data_in[81][0]}} & 8'h2d)^
({{8{data_in[81][1]}} & 8'h5a)^
({{8{data_in[81][2]}} & 8'hb4)^
({{8{data_in[81][3]}} & 8'h43)^
({{8{data_in[81][4]}} & 8'h86})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[81][5]}} & 8'h27)^
({{8{data_in[81][6]}} & 8'h4e)^
({{8{data_in[81][7]}} & 8'h9c)^
({{8{data_in[82][0]}} & 8'h24)^
({{8{data_in[82][1]}} & 8'h48)^
({{8{data_in[82][2]}} & 8'h90)^
({{8{data_in[82][3]}} & 8'hb)^
({{8{data_in[82][4]}} & 8'h16)^
({{8{data_in[82][5]}} & 8'h2c)^
({{8{data_in[82][6]}} & 8'h58)^
({{8{data_in[82][7]}} & 8'hb0)^
({{8{data_in[83][0]}} & 8'hba)^
({{8{data_in[83][1]}} & 8'h5f)^
({{8{data_in[83][2]}} & 8'hbe)^
({{8{data_in[83][3]}} & 8'h57)^
({{8{data_in[83][4]}} & 8'hae)^
({{8{data_in[83][5]}} & 8'h77)^
({{8{data_in[83][6]}} & 8'hee)^
({{8{data_in[83][7]}} & 8'hf7)^
({{8{data_in[84][0]}} & 8'h2b)^
({{8{data_in[84][1]}} & 8'h56)^
({{8{data_in[84][2]}} & 8'hac)^
({{8{data_in[84][3]}} & 8'h73)^
({{8{data_in[84][4]}} & 8'he6)^
({{8{data_in[84][5]}} & 8'he7)^
({{8{data_in[84][6]}} & 8'he5)^
({{8{data_in[84][7]}} & 8'he1)^
({{8{data_in[85][0]}} & 8'hb0)^
({{8{data_in[85][1]}} & 8'h4b)^
({{8{data_in[85][2]}} & 8'h96)^
({{8{data_in[85][3]}} & 8'h7)^
({{8{data_in[85][4]}} & 8'he)^
({{8{data_in[85][5]}} & 8'h1c)^
({{8{data_in[85][6]}} & 8'h38)^
({{8{data_in[85][7]}} & 8'h70)^
({{8{data_in[86][0]}} & 8'he1)^
({{8{data_in[86][1]}} & 8'he9)^
({{8{data_in[86][2]}} & 8'hf9)^
({{8{data_in[86][3]}} & 8'hd9)^
({{8{data_in[86][4]}} & 8'h99)^
({{8{data_in[86][5]}} & 8'h19)^
({{8{data_in[86][6]}} & 8'h32)^
({{8{data_in[86][7]}} & 8'h64)^
({{8{data_in[87][0]}} & 8'h57)^
({{8{data_in[87][1]}} & 8'hae)^
({{8{data_in[87][2]}} & 8'h77)^
({{8{data_in[87][3]}} & 8'hee)^
({{8{data_in[87][4]}} & 8'hf7)^
({{8{data_in[87][5]}} & 8'hc5)^
({{8{data_in[87][6]}} & 8'ha1)^
({{8{data_in[87][7]}} & 8'h69)^
({{8{data_in[88][0]}} & 8'h1c)^
({{8{data_in[88][1]}} & 8'h38)^
({{8{data_in[88][2]}} & 8'h70)^
({{8{data_in[88][3]}} & 8'he0)^
({{8{data_in[88][4]}} & 8'heb)^
({{8{data_in[88][5]}} & 8'hfd)^
({{8{data_in[88][6]}} & 8'hd1)^
({{8{data_in[88][7]}} & 8'h89)^
({{8{data_in[89][0]}} & 8'h54)^
({{8{data_in[89][1]}} & 8'ha8)^
({{8{data_in[89][2]}} & 8'h7b)^
({{8{data_in[89][3]}} & 8'hf6)^
({{8{data_in[89][4]}} & 8'hc7)^
({{8{data_in[89][5]}} & 8'ha5})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[89][6]}} & 8'h61)^
({{8{data_in[89][7]}} & 8'hc2)^
({{8{data_in[90][0]}} & 8'h55)^
({{8{data_in[90][1]}} & 8'haa)^
({{8{data_in[90][2]}} & 8'h7f)^
({{8{data_in[90][3]}} & 8'hfe)^
({{8{data_in[90][4]}} & 8'hd7)^
({{8{data_in[90][5]}} & 8'h85)^
({{8{data_in[90][6]}} & 8'h21)^
({{8{data_in[90][7]}} & 8'h42)^
({{8{data_in[91][0]}} & 8'h5a)^
({{8{data_in[91][1]}} & 8'hb4)^
({{8{data_in[91][2]}} & 8'h43)^
({{8{data_in[91][3]}} & 8'h86)^
({{8{data_in[91][4]}} & 8'h27)^
({{8{data_in[91][5]}} & 8'h4e)^
({{8{data_in[91][6]}} & 8'h9c)^
({{8{data_in[91][7]}} & 8'h13)^
({{8{data_in[92][0]}} & 8'ha3)^
({{8{data_in[92][1]}} & 8'h6d)^
({{8{data_in[92][2]}} & 8'hda)^
({{8{data_in[92][3]}} & 8'h9f)^
({{8{data_in[92][4]}} & 8'h15)^
({{8{data_in[92][5]}} & 8'h2a)^
({{8{data_in[92][6]}} & 8'h54)^
({{8{data_in[92][7]}} & 8'ha8)^
({{8{data_in[93][0]}} & 8'hd6)^
({{8{data_in[93][1]}} & 8'h87)^
({{8{data_in[93][2]}} & 8'h25)^
({{8{data_in[93][3]}} & 8'h4a)^
({{8{data_in[93][4]}} & 8'h94)^
({{8{data_in[93][5]}} & 8'h3)^
({{8{data_in[93][6]}} & 8'h6)^
({{8{data_in[93][7]}} & 8'hc)^
({{8{data_in[94][0]}} & 8'h7c)^
({{8{data_in[94][1]}} & 8'hf8)^
({{8{data_in[94][2]}} & 8'hdb)^
({{8{data_in[94][3]}} & 8'h9d)^
({{8{data_in[94][4]}} & 8'h11)^
({{8{data_in[94][5]}} & 8'h22)^
({{8{data_in[94][6]}} & 8'h44)^
({{8{data_in[94][7]}} & 8'h88)^
({{8{data_in[95][0]}} & 8'he7)^
({{8{data_in[95][1]}} & 8'he5)^
({{8{data_in[95][2]}} & 8'he1)^
({{8{data_in[95][3]}} & 8'he9)^
({{8{data_in[95][4]}} & 8'hf9)^
({{8{data_in[95][5]}} & 8'hd9)^
({{8{data_in[95][6]}} & 8'h99)^
({{8{data_in[95][7]}} & 8'h19)^
({{8{data_in[96][0]}} & 8'h4f)^
({{8{data_in[96][1]}} & 8'h9e)^
({{8{data_in[96][2]}} & 8'h17)^
({{8{data_in[96][3]}} & 8'h2e)^
({{8{data_in[96][4]}} & 8'h5c)^
({{8{data_in[96][5]}} & 8'hb8)^
({{8{data_in[96][6]}} & 8'h5b)^
({{8{data_in[96][7]}} & 8'hb6)^
({{8{data_in[97][0]}} & 8'h93)^
({{8{data_in[97][1]}} & 8'hd)^
({{8{data_in[97][2]}} & 8'h1a)^
({{8{data_in[97][3]}} & 8'h34)^
({{8{data_in[97][4]}} & 8'h68)^
({{8{data_in[97][5]}} & 8'hd0)^
({{8{data_in[97][6]}} & 8'h8b)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[97][7]}} & 8'h3d)^
({{8{data_in[98][0]}} & 8'h66)^
({{8{data_in[98][1]}} & 8'hcc)^
({{8{data_in[98][2]}} & 8'hb3)^
({{8{data_in[98][3]}} & 8'h4d)^
({{8{data_in[98][4]}} & 8'h9a)^
({{8{data_in[98][5]}} & 8'h1f)^
({{8{data_in[98][6]}} & 8'h3e)^
({{8{data_in[98][7]}} & 8'h7c)^
({{8{data_in[99][0]}} & 8'h3b)^
({{8{data_in[99][1]}} & 8'h76)^
({{8{data_in[99][2]}} & 8'hec)^
({{8{data_in[99][3]}} & 8'hf3)^
({{8{data_in[99][4]}} & 8'hcd)^
({{8{data_in[99][5]}} & 8'hb1)^
({{8{data_in[99][6]}} & 8'h49)^
({{8{data_in[99][7]}} & 8'h92)^
({{8{data_in[100][0]}} & 8'h4f)^
({{8{data_in[100][1]}} & 8'h9e)^
({{8{data_in[100][2]}} & 8'h17)^
({{8{data_in[100][3]}} & 8'h2e)^
({{8{data_in[100][4]}} & 8'h5c)^
({{8{data_in[100][5]}} & 8'hb8)^
({{8{data_in[100][6]}} & 8'h5b)^
({{8{data_in[100][7]}} & 8'hb6)^
({{8{data_in[101][0]}} & 8'h15)^
({{8{data_in[101][1]}} & 8'h2a)^
({{8{data_in[101][2]}} & 8'h54)^
({{8{data_in[101][3]}} & 8'ha8)^
({{8{data_in[101][4]}} & 8'h7b)^
({{8{data_in[101][5]}} & 8'hf6)^
({{8{data_in[101][6]}} & 8'hc7)^
({{8{data_in[101][7]}} & 8'ha5)^
({{8{data_in[102][0]}} & 8'h14)^
({{8{data_in[102][1]}} & 8'h28)^
({{8{data_in[102][2]}} & 8'h50)^
({{8{data_in[102][3]}} & 8'ha0)^
({{8{data_in[102][4]}} & 8'h6b)^
({{8{data_in[102][5]}} & 8'hd6)^
({{8{data_in[102][6]}} & 8'h87)^
({{8{data_in[102][7]}} & 8'h25)^
({{8{data_in[103][0]}} & 8'h13)^
({{8{data_in[103][1]}} & 8'h26)^
({{8{data_in[103][2]}} & 8'h4c)^
({{8{data_in[103][3]}} & 8'h98)^
({{8{data_in[103][4]}} & 8'h1b)^
({{8{data_in[103][5]}} & 8'h36)^
({{8{data_in[103][6]}} & 8'h6c)^
({{8{data_in[103][7]}} & 8'hd8)^
({{8{data_in[104][0]}} & 8'h6b)^
({{8{data_in[104][1]}} & 8'hd6)^
({{8{data_in[104][2]}} & 8'h87)^
({{8{data_in[104][3]}} & 8'h25)^
({{8{data_in[104][4]}} & 8'h4a)^
({{8{data_in[104][5]}} & 8'h94)^
({{8{data_in[104][6]}} & 8'h3)^
({{8{data_in[104][7]}} & 8'h6)^
({{8{data_in[105][0]}} & 8'h60)^
({{8{data_in[105][1]}} & 8'hc0)^
({{8{data_in[105][2]}} & 8'hab)^
({{8{data_in[105][3]}} & 8'h7d)^
({{8{data_in[105][4]}} & 8'hfa)^
({{8{data_in[105][5]}} & 8'hdf)^
({{8{data_in[105][6]}} & 8'h95)^
({{8{data_in[105][7]}} & 8'h1)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[106][0]}} & 8'h1a)^
({{8{data_in[106][1]}} & 8'h34)^
({{8{data_in[106][2]}} & 8'h68)^
({{8{data_in[106][3]}} & 8'hd0)^
({{8{data_in[106][4]}} & 8'h8b)^
({{8{data_in[106][5]}} & 8'h3d)^
({{8{data_in[106][6]}} & 8'h7a)^
({{8{data_in[106][7]}} & 8'hf4)^
({{8{data_in[107][0]}} & 8'h25)^
({{8{data_in[107][1]}} & 8'h4a)^
({{8{data_in[107][2]}} & 8'h94)^
({{8{data_in[107][3]}} & 8'h3)^
({{8{data_in[107][4]}} & 8'h6)^
({{8{data_in[107][5]}} & 8'hc)^
({{8{data_in[107][6]}} & 8'h18)^
({{8{data_in[107][7]}} & 8'h30)^
({{8{data_in[108][0]}} & 8'h79)^
({{8{data_in[108][1]}} & 8'hf2)^
({{8{data_in[108][2]}} & 8'hcf)^
({{8{data_in[108][3]}} & 8'hb5)^
({{8{data_in[108][4]}} & 8'h41)^
({{8{data_in[108][5]}} & 8'h82)^
({{8{data_in[108][6]}} & 8'h2f)^
({{8{data_in[108][7]}} & 8'h5e)^
({{8{data_in[109][0]}} & 8'h6b)^
({{8{data_in[109][1]}} & 8'hd6)^
({{8{data_in[109][2]}} & 8'h87)^
({{8{data_in[109][3]}} & 8'h25)^
({{8{data_in[109][4]}} & 8'h4a)^
({{8{data_in[109][5]}} & 8'h94)^
({{8{data_in[109][6]}} & 8'h3)^
({{8{data_in[109][7]}} & 8'h6)^
({{8{data_in[110][0]}} & 8'hd3)^
({{8{data_in[110][1]}} & 8'h8d)^
({{8{data_in[110][2]}} & 8'h31)^
({{8{data_in[110][3]}} & 8'h62)^
({{8{data_in[110][4]}} & 8'hc4)^
({{8{data_in[110][5]}} & 8'ha3)^
({{8{data_in[110][6]}} & 8'h6d)^
({{8{data_in[110][7]}} & 8'hda)^
({{8{data_in[111][0]}} & 8'h79)^
({{8{data_in[111][1]}} & 8'hf2)^
({{8{data_in[111][2]}} & 8'hcf)^
({{8{data_in[111][3]}} & 8'hb5)^
({{8{data_in[111][4]}} & 8'h41)^
({{8{data_in[111][5]}} & 8'h82)^
({{8{data_in[111][6]}} & 8'h2f)^
({{8{data_in[111][7]}} & 8'h5e)^
({{8{data_in[112][0]}} & 8'h13)^
({{8{data_in[112][1]}} & 8'h26)^
({{8{data_in[112][2]}} & 8'h4c)^
({{8{data_in[112][3]}} & 8'h98)^
({{8{data_in[112][4]}} & 8'h1b)^
({{8{data_in[112][5]}} & 8'h36)^
({{8{data_in[112][6]}} & 8'h6c)^
({{8{data_in[112][7]}} & 8'hd8)^
({{8{data_in[113][0]}} & 8'h6)^
({{8{data_in[113][1]}} & 8'hc)^
({{8{data_in[113][2]}} & 8'h18)^
({{8{data_in[113][3]}} & 8'h30)^
({{8{data_in[113][4]}} & 8'h60)^
({{8{data_in[113][5]}} & 8'hc0)^
({{8{data_in[113][6]}} & 8'hab)^
({{8{data_in[113][7]}} & 8'h7d)^
({{8{data_in[114][0]}} & 8'hff)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[114][1]}} & 8'hd5)^
({{8{data_in[114][2]}} & 8'h81)^
({{8{data_in[114][3]}} & 8'h29)^
({{8{data_in[114][4]}} & 8'h52)^
({{8{data_in[114][5]}} & 8'ha4)^
({{8{data_in[114][6]}} & 8'h63)^
({{8{data_in[114][7]}} & 8'hc6)^
({{8{data_in[115][0]}} & 8'h88)^
({{8{data_in[115][1]}} & 8'h3b)^
({{8{data_in[115][2]}} & 8'h76)^
({{8{data_in[115][3]}} & 8'hec)^
({{8{data_in[115][4]}} & 8'hf3)^
({{8{data_in[115][5]}} & 8'hcd)^
({{8{data_in[115][6]}} & 8'hb1)^
({{8{data_in[115][7]}} & 8'h49)^
({{8{data_in[116][0]}} & 8'hbc)^
({{8{data_in[116][1]}} & 8'h53)^
({{8{data_in[116][2]}} & 8'ha6)^
({{8{data_in[116][3]}} & 8'h67)^
({{8{data_in[116][4]}} & 8'hce)^
({{8{data_in[116][5]}} & 8'hb7)^
({{8{data_in[116][6]}} & 8'h45)^
({{8{data_in[116][7]}} & 8'h8a)^
({{8{data_in[117][0]}} & 8'h93)^
({{8{data_in[117][1]}} & 8'hd)^
({{8{data_in[117][2]}} & 8'h1a)^
({{8{data_in[117][3]}} & 8'h34)^
({{8{data_in[117][4]}} & 8'h68)^
({{8{data_in[117][5]}} & 8'hd0)^
({{8{data_in[117][6]}} & 8'h8b)^
({{8{data_in[117][7]}} & 8'h3d)^
({{8{data_in[118][0]}} & 8'h5a)^
({{8{data_in[118][1]}} & 8'hb4)^
({{8{data_in[118][2]}} & 8'h43)^
({{8{data_in[118][3]}} & 8'h86)^
({{8{data_in[118][4]}} & 8'h27)^
({{8{data_in[118][5]}} & 8'h4e)^
({{8{data_in[118][6]}} & 8'h9c)^
({{8{data_in[118][7]}} & 8'h13)^
({{8{data_in[119][0]}} & 8'h6d)^
({{8{data_in[119][1]}} & 8'hda)^
({{8{data_in[119][2]}} & 8'h9f)^
({{8{data_in[119][3]}} & 8'h15)^
({{8{data_in[119][4]}} & 8'h2a)^
({{8{data_in[119][5]}} & 8'h54)^
({{8{data_in[119][6]}} & 8'ha8)^
({{8{data_in[119][7]}} & 8'h7b)^
({{8{data_in[120][0]}} & 8'h73)^
({{8{data_in[120][1]}} & 8'he6)^
({{8{data_in[120][2]}} & 8'he7)^
({{8{data_in[120][3]}} & 8'he5)^
({{8{data_in[120][4]}} & 8'he1)^
({{8{data_in[120][5]}} & 8'he9)^
({{8{data_in[120][6]}} & 8'hf9)^
({{8{data_in[120][7]}} & 8'hd9)^
({{8{data_in[121][0]}} & 8'hb8)^
({{8{data_in[121][1]}} & 8'h5b)^
({{8{data_in[121][2]}} & 8'hb6)^
({{8{data_in[121][3]}} & 8'h47)^
({{8{data_in[121][4]}} & 8'h8e)^
({{8{data_in[121][5]}} & 8'h37)^
({{8{data_in[121][6]}} & 8'h6e)^
({{8{data_in[121][7]}} & 8'hdc)^
({{8{data_in[122][0]}} & 8'h5b)^
({{8{data_in[122][1]}} & 8'hb6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[122][2]}} & 8'h47)^
({{8{data_in[122][3]}} & 8'h8e)^
({{8{data_in[122][4]}} & 8'h37)^
({{8{data_in[122][5]}} & 8'h6e)^
({{8{data_in[122][6]}} & 8'hdcc)^
({{8{data_in[122][7]}} & 8'h93)^
({{8{data_in[123][0]}} & 8'h45)^
({{8{data_in[123][1]}} & 8'h8a)^
({{8{data_in[123][2]}} & 8'h3f)^
({{8{data_in[123][3]}} & 8'h7e)^
({{8{data_in[123][4]}} & 8'hfc)^
({{8{data_in[123][5]}} & 8'hd3)^
({{8{data_in[123][6]}} & 8'h8d)^
({{8{data_in[123][7]}} & 8'h31)^
({{8{data_in[124][0]}} & 8'hac)^
({{8{data_in[124][1]}} & 8'h73)^
({{8{data_in[124][2]}} & 8'he6)^
({{8{data_in[124][3]}} & 8'he7)^
({{8{data_in[124][4]}} & 8'he5)^
({{8{data_in[124][5]}} & 8'he1)^
({{8{data_in[124][6]}} & 8'he9)^
({{8{data_in[124][7]}} & 8'hf9)^
({{8{data_in[125][0]}} & 8'h4b)^
({{8{data_in[125][1]}} & 8'h96)^
({{8{data_in[125][2]}} & 8'h7)^
({{8{data_in[125][3]}} & 8'he)^
({{8{data_in[125][4]}} & 8'h1c)^
({{8{data_in[125][5]}} & 8'h38)^
({{8{data_in[125][6]}} & 8'h70)^
({{8{data_in[125][7]}} & 8'he0)^
({{8{data_in[126][0]}} & 8'hf5)^
({{8{data_in[126][1]}} & 8'hc1)^
({{8{data_in[126][2]}} & 8'ha9)^
({{8{data_in[126][3]}} & 8'h79)^
({{8{data_in[126][4]}} & 8'hf2)^
({{8{data_in[126][5]}} & 8'hcf)^
({{8{data_in[126][6]}} & 8'hb5)^
({{8{data_in[126][7]}} & 8'h41)^
({{8{data_in[127][0]}} & 8'h33)^
({{8{data_in[127][1]}} & 8'h66)^
({{8{data_in[127][2]}} & 8'hcc)^
({{8{data_in[127][3]}} & 8'hb3)^
({{8{data_in[127][4]}} & 8'h4d)^
({{8{data_in[127][5]}} & 8'h9a)^
({{8{data_in[127][6]}} & 8'h1f)^
({{8{data_in[127][7]}} & 8'h3e)^
({{8{data_in[128][0]}} & 8'h86)^
({{8{data_in[128][1]}} & 8'h27)^
({{8{data_in[128][2]}} & 8'h4e)^
({{8{data_in[128][3]}} & 8'h9c)^
({{8{data_in[128][4]}} & 8'h13)^
({{8{data_in[128][5]}} & 8'h26)^
({{8{data_in[128][6]}} & 8'h4c)^
({{8{data_in[128][7]}} & 8'h98)^
({{8{data_in[129][0]}} & 8'h5b)^
({{8{data_in[129][1]}} & 8'hb6)^
({{8{data_in[129][2]}} & 8'h47)^
({{8{data_in[129][3]}} & 8'h8e)^
({{8{data_in[129][4]}} & 8'h37)^
({{8{data_in[129][5]}} & 8'h6e)^
({{8{data_in[129][6]}} & 8'hdcc)^
({{8{data_in[129][7]}} & 8'h93)^
({{8{data_in[130][0]}} & 8'ha0)^
({{8{data_in[130][1]}} & 8'h6b)^
({{8{data_in[130][2]}} & 8'hd6)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[130][3]}} & 8'h87)^
({{8{data_in[130][4]}} & 8'h25)^
({{8{data_in[130][5]}} & 8'h4a)^
({{8{data_in[130][6]}} & 8'h94)^
({{8{data_in[130][7]}} & 8'h3)^
({{8{data_in[131][0]}} & 8'h5e)^
({{8{data_in[131][1]}} & 8'hbc)^
({{8{data_in[131][2]}} & 8'h53)^
({{8{data_in[131][3]}} & 8'ha6)^
({{8{data_in[131][4]}} & 8'h67)^
({{8{data_in[131][5]}} & 8'hce)^
({{8{data_in[131][6]}} & 8'hb7)^
({{8{data_in[131][7]}} & 8'h45)^
({{8{data_in[132][0]}} & 8'he1)^
({{8{data_in[132][1]}} & 8'he9)^
({{8{data_in[132][2]}} & 8'hf9)^
({{8{data_in[132][3]}} & 8'hd9)^
({{8{data_in[132][4]}} & 8'h99)^
({{8{data_in[132][5]}} & 8'h19)^
({{8{data_in[132][6]}} & 8'h32)^
({{8{data_in[132][7]}} & 8'h64)^
({{8{data_in[133][0]}} & 8'h50)^
({{8{data_in[133][1]}} & 8'ha0)^
({{8{data_in[133][2]}} & 8'h6b)^
({{8{data_in[133][3]}} & 8'hd6)^
({{8{data_in[133][4]}} & 8'h87)^
({{8{data_in[133][5]}} & 8'h25)^
({{8{data_in[133][6]}} & 8'h4a)^
({{8{data_in[133][7]}} & 8'h94)^
({{8{data_in[134][0]}} & 8'h1e)^
({{8{data_in[134][1]}} & 8'h3c)^
({{8{data_in[134][2]}} & 8'h78)^
({{8{data_in[134][3]}} & 8'hf0)^
({{8{data_in[134][4]}} & 8'hcb)^
({{8{data_in[134][5]}} & 8'hbd)^
({{8{data_in[134][6]}} & 8'h51)^
({{8{data_in[134][7]}} & 8'ha2)^
({{8{data_in[135][0]}} & 8'h55)^
({{8{data_in[135][1]}} & 8'haa)^
({{8{data_in[135][2]}} & 8'h7f)^
({{8{data_in[135][3]}} & 8'hfe)^
({{8{data_in[135][4]}} & 8'hd7)^
({{8{data_in[135][5]}} & 8'h85)^
({{8{data_in[135][6]}} & 8'h21)^
({{8{data_in[135][7]}} & 8'h42)^
({{8{data_in[136][0]}} & 8'h83)^
({{8{data_in[136][1]}} & 8'h2d)^
({{8{data_in[136][2]}} & 8'h5a)^
({{8{data_in[136][3]}} & 8'hb4)^
({{8{data_in[136][4]}} & 8'h43)^
({{8{data_in[136][5]}} & 8'h86)^
({{8{data_in[136][6]}} & 8'h27)^
({{8{data_in[136][7]}} & 8'h4e)^
({{8{data_in[137][0]}} & 8'h60)^
({{8{data_in[137][1]}} & 8'hc0)^
({{8{data_in[137][2]}} & 8'hab)^
({{8{data_in[137][3]}} & 8'h7d)^
({{8{data_in[137][4]}} & 8'hfa)^
({{8{data_in[137][5]}} & 8'hdf)^
({{8{data_in[137][6]}} & 8'h95)^
({{8{data_in[137][7]}} & 8'h1)^
({{8{data_in[138][0]}} & 8'h49)^
({{8{data_in[138][1]}} & 8'h92)^
({{8{data_in[138][2]}} & 8'hf)^
({{8{data_in[138][3]}} & 8'h1e)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[138][4]}} & 8'h3c)^
({{8{data_in[138][5]}} & 8'h78)^
({{8{data_in[138][6]}} & 8'hf0)^
({{8{data_in[138][7]}} & 8'hcb)^
({{8{data_in[139][0]}} & 8'hed)^
({{8{data_in[139][1]}} & 8'hf1)^
({{8{data_in[139][2]}} & 8'hc9)^
({{8{data_in[139][3]}} & 8'hb9)^
({{8{data_in[139][4]}} & 8'h59)^
({{8{data_in[139][5]}} & 8'hb2)^
({{8{data_in[139][6]}} & 8'h4f)^
({{8{data_in[139][7]}} & 8'h9e)^
({{8{data_in[140][0]}} & 8'hd9)^
({{8{data_in[140][1]}} & 8'h99)^
({{8{data_in[140][2]}} & 8'h19)^
({{8{data_in[140][3]}} & 8'h32)^
({{8{data_in[140][4]}} & 8'h64)^
({{8{data_in[140][5]}} & 8'hc8)^
({{8{data_in[140][6]}} & 8'hbb)^
({{8{data_in[140][7]}} & 8'h5d)^
({{8{data_in[141][0]}} & 8'h41)^
({{8{data_in[141][1]}} & 8'h82)^
({{8{data_in[141][2]}} & 8'h2f)^
({{8{data_in[141][3]}} & 8'h5e)^
({{8{data_in[141][4]}} & 8'hbc)^
({{8{data_in[141][5]}} & 8'h53)^
({{8{data_in[141][6]}} & 8'ha6)^
({{8{data_in[141][7]}} & 8'h67)^
({{8{data_in[142][0]}} & 8'h86)^
({{8{data_in[142][1]}} & 8'h27)^
({{8{data_in[142][2]}} & 8'h4e)^
({{8{data_in[142][3]}} & 8'h9c)^
({{8{data_in[142][4]}} & 8'h13)^
({{8{data_in[142][5]}} & 8'h26)^
({{8{data_in[142][6]}} & 8'h4c)^
({{8{data_in[142][7]}} & 8'h98)^
({{8{data_in[143][0]}} & 8'h2)^
({{8{data_in[143][1]}} & 8'h4)^
({{8{data_in[143][2]}} & 8'h8)^
({{8{data_in[143][3]}} & 8'h10)^
({{8{data_in[143][4]}} & 8'h20)^
({{8{data_in[143][5]}} & 8'h40)^
({{8{data_in[143][6]}} & 8'h80)^
({{8{data_in[143][7]}} & 8'h2b)^
({{8{data_in[144][0]}} & 8'h8)^
({{8{data_in[144][1]}} & 8'h10)^
({{8{data_in[144][2]}} & 8'h20)^
({{8{data_in[144][3]}} & 8'h40)^
({{8{data_in[144][4]}} & 8'h80)^
({{8{data_in[144][5]}} & 8'h2b)^
({{8{data_in[144][6]}} & 8'h56)^
({{8{data_in[144][7]}} & 8'hac)^
({{8{data_in[145][0]}} & 8'hb8)^
({{8{data_in[145][1]}} & 8'h5b)^
({{8{data_in[145][2]}} & 8'hb6)^
({{8{data_in[145][3]}} & 8'h47)^
({{8{data_in[145][4]}} & 8'h8e)^
({{8{data_in[145][5]}} & 8'h37)^
({{8{data_in[145][6]}} & 8'h6e)^
({{8{data_in[145][7]}} & 8'hdc)^
({{8{data_in[146][0]}} & 8'h1)^
({{8{data_in[146][1]}} & 8'h2)^
({{8{data_in[146][2]}} & 8'h4)^
({{8{data_in[146][3]}} & 8'h8)^
({{8{data_in[146][4]}} & 8'h10)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[146][5]}} & 8'h20)^
({{8{data_in[146][6]}} & 8'h40)^
({{8{data_in[146][7]}} & 8'h80)^
({{8{data_in[147][0]}} & 8'h1d)^
({{8{data_in[147][1]}} & 8'h3a)^
({{8{data_in[147][2]}} & 8'h74)^
({{8{data_in[147][3]}} & 8'he8)^
({{8{data_in[147][4]}} & 8'hfb)^
({{8{data_in[147][5]}} & 8'hdd)^
({{8{data_in[147][6]}} & 8'h91)^
({{8{data_in[147][7]}} & 8'h9)^
({{8{data_in[148][0]}} & 8'h74)^
({{8{data_in[148][1]}} & 8'he8)^
({{8{data_in[148][2]}} & 8'hfb)^
({{8{data_in[148][3]}} & 8'hdd)^
({{8{data_in[148][4]}} & 8'h91)^
({{8{data_in[148][5]}} & 8'h9)^
({{8{data_in[148][6]}} & 8'h12)^
({{8{data_in[148][7]}} & 8'h24)^
({{8{data_in[149][0]}} & 8'h98)^
({{8{data_in[149][1]}} & 8'h1b)^
({{8{data_in[149][2]}} & 8'h36)^
({{8{data_in[149][3]}} & 8'h6c)^
({{8{data_in[149][4]}} & 8'hd8)^
({{8{data_in[149][5]}} & 8'h9b)^
({{8{data_in[149][6]}} & 8'h1d)^
({{8{data_in[149][7]}} & 8'h3a)^
({{8{data_in[150][0]}} & 8'hf5)^
({{8{data_in[150][1]}} & 8'hc1)^
({{8{data_in[150][2]}} & 8'ha9)^
({{8{data_in[150][3]}} & 8'h79)^
({{8{data_in[150][4]}} & 8'hf2)^
({{8{data_in[150][5]}} & 8'hcf)^
({{8{data_in[150][6]}} & 8'hb5)^
({{8{data_in[150][7]}} & 8'h41)^
({{8{data_in[151][0]}} & 8'hf3)^
({{8{data_in[151][1]}} & 8'hcd)^
({{8{data_in[151][2]}} & 8'hb1)^
({{8{data_in[151][3]}} & 8'h49)^
({{8{data_in[151][4]}} & 8'h92)^
({{8{data_in[151][5]}} & 8'hf)^
({{8{data_in[151][6]}} & 8'h1e)^
({{8{data_in[151][7]}} & 8'h3c)^
({{8{data_in[152][0]}} & 8'hfa)^
({{8{data_in[152][1]}} & 8'hdf)^
({{8{data_in[152][2]}} & 8'h95)^
({{8{data_in[152][3]}} & 8'h1)^
({{8{data_in[152][4]}} & 8'h2)^
({{8{data_in[152][5]}} & 8'h4)^
({{8{data_in[152][6]}} & 8'h8)^
({{8{data_in[152][7]}} & 8'h10)^
({{8{data_in[153][0]}} & 8'he0)^
({{8{data_in[153][1]}} & 8'heb)^
({{8{data_in[153][2]}} & 8'hfd)^
({{8{data_in[153][3]}} & 8'hd1)^
({{8{data_in[153][4]}} & 8'h89)^
({{8{data_in[153][5]}} & 8'h39)^
({{8{data_in[153][6]}} & 8'h72)^
({{8{data_in[153][7]}} & 8'he4)^
({{8{data_in[154][0]}} & 8'h6)^
({{8{data_in[154][1]}} & 8'hc)^
({{8{data_in[154][2]}} & 8'h18)^
({{8{data_in[154][3]}} & 8'h30)^
({{8{data_in[154][4]}} & 8'h60)^
({{8{data_in[154][5]}} & 8'hc0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[154][6]}} & 8'hab)^
({{8{data_in[154][7]}} & 8'h7d)^
({{8{data_in[155][0]}} & 8'he3)^
({{8{data_in[155][1]}} & 8'hed)^
({{8{data_in[155][2]}} & 8'hf1)^
({{8{data_in[155][3]}} & 8'hc9)^
({{8{data_in[155][4]}} & 8'hb9)^
({{8{data_in[155][5]}} & 8'h59)^
({{8{data_in[155][6]}} & 8'hb2)^
({{8{data_in[155][7]}} & 8'h4f)^
({{8{data_in[156][0]}} & 8'hb8)^
({{8{data_in[156][1]}} & 8'h5b)^
({{8{data_in[156][2]}} & 8'hb6)^
({{8{data_in[156][3]}} & 8'h47)^
({{8{data_in[156][4]}} & 8'h8e)^
({{8{data_in[156][5]}} & 8'h37)^
({{8{data_in[156][6]}} & 8'h6e)^
({{8{data_in[156][7]}} & 8'hdc)^
({{8{data_in[157][0]}} & 8'hff)^
({{8{data_in[157][1]}} & 8'hd5)^
({{8{data_in[157][2]}} & 8'h81)^
({{8{data_in[157][3]}} & 8'h29)^
({{8{data_in[157][4]}} & 8'h52)^
({{8{data_in[157][5]}} & 8'ha4)^
({{8{data_in[157][6]}} & 8'h63)^
({{8{data_in[157][7]}} & 8'hc6)^
({{8{data_in[158][0]}} & 8'he8)^
({{8{data_in[158][1]}} & 8'hfb)^
({{8{data_in[158][2]}} & 8'hdd)^
({{8{data_in[158][3]}} & 8'h91)^
({{8{data_in[158][4]}} & 8'h9)^
({{8{data_in[158][5]}} & 8'h12)^
({{8{data_in[158][6]}} & 8'h24)^
({{8{data_in[158][7]}} & 8'h48)^
({{8{data_in[159][0]}} & 8'h17)^
({{8{data_in[159][1]}} & 8'h2e)^
({{8{data_in[159][2]}} & 8'h5c)^
({{8{data_in[159][3]}} & 8'hb8)^
({{8{data_in[159][4]}} & 8'h5b)^
({{8{data_in[159][5]}} & 8'hb6)^
({{8{data_in[159][6]}} & 8'h47)^
({{8{data_in[159][7]}} & 8'h8e)^
({{8{data_in[160][0]}} & 8'h9c)^
({{8{data_in[160][1]}} & 8'h13)^
({{8{data_in[160][2]}} & 8'h26)^
({{8{data_in[160][3]}} & 8'h4c)^
({{8{data_in[160][4]}} & 8'h98)^
({{8{data_in[160][5]}} & 8'h1b)^
({{8{data_in[160][6]}} & 8'h36)^
({{8{data_in[160][7]}} & 8'h6c)^
({{8{data_in[161][0]}} & 8'h7a)^
({{8{data_in[161][1]}} & 8'hf4)^
({{8{data_in[161][2]}} & 8'hc3)^
({{8{data_in[161][3]}} & 8'had)^
({{8{data_in[161][4]}} & 8'h71)^
({{8{data_in[161][5]}} & 8'he2)^
({{8{data_in[161][6]}} & 8'hef)^
({{8{data_in[161][7]}} & 8'hf5)^
({{8{data_in[162][0]}} & 8'h59)^
({{8{data_in[162][1]}} & 8'hb2)^
({{8{data_in[162][2]}} & 8'h4f)^
({{8{data_in[162][3]}} & 8'h9e)^
({{8{data_in[162][4]}} & 8'h17)^
({{8{data_in[162][5]}} & 8'h2e)^
({{8{data_in[162][6]}} & 8'h5c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[162][7]}} & 8'hb8)^
({{8{data_in[163][0]}} & 8'h1f)^
({{8{data_in[163][1]}} & 8'h3e)^
({{8{data_in[163][2]}} & 8'h7c)^
({{8{data_in[163][3]}} & 8'hf8)^
({{8{data_in[163][4]}} & 8'hdb)^
({{8{data_in[163][5]}} & 8'h9d)^
({{8{data_in[163][6]}} & 8'h11)^
({{8{data_in[163][7]}} & 8'h22)^
({{8{data_in[164][0]}} & 8'hbf)^
({{8{data_in[164][1]}} & 8'h55)^
({{8{data_in[164][2]}} & 8'haa)^
({{8{data_in[164][3]}} & 8'h7f)^
({{8{data_in[164][4]}} & 8'hfe)^
({{8{data_in[164][5]}} & 8'hd7)^
({{8{data_in[164][6]}} & 8'h85)^
({{8{data_in[164][7]}} & 8'h21)^
({{8{data_in[165][0]}} & 8'hb7)^
({{8{data_in[165][1]}} & 8'h45)^
({{8{data_in[165][2]}} & 8'h8a)^
({{8{data_in[165][3]}} & 8'h3f)^
({{8{data_in[165][4]}} & 8'h7e)^
({{8{data_in[165][5]}} & 8'hfc)^
({{8{data_in[165][6]}} & 8'hd3)^
({{8{data_in[165][7]}} & 8'h8d)^
({{8{data_in[166][0]}} & 8'had)^
({{8{data_in[166][1]}} & 8'h71)^
({{8{data_in[166][2]}} & 8'he2)^
({{8{data_in[166][3]}} & 8'hef)^
({{8{data_in[166][4]}} & 8'hf5)^
({{8{data_in[166][5]}} & 8'hc1)^
({{8{data_in[166][6]}} & 8'ha9)^
({{8{data_in[166][7]}} & 8'h79)^
({{8{data_in[167][0]}} & 8'h71)^
({{8{data_in[167][1]}} & 8'he2)^
({{8{data_in[167][2]}} & 8'hef)^
({{8{data_in[167][3]}} & 8'hf5)^
({{8{data_in[167][4]}} & 8'hc1)^
({{8{data_in[167][5]}} & 8'ha9)^
({{8{data_in[167][6]}} & 8'h79)^
({{8{data_in[167][7]}} & 8'hf2)^
({{8{data_in[168][0]}} & 8'h81)^
({{8{data_in[168][1]}} & 8'h29)^
({{8{data_in[168][2]}} & 8'h52)^
({{8{data_in[168][3]}} & 8'ha4)^
({{8{data_in[168][4]}} & 8'h63)^
({{8{data_in[168][5]}} & 8'hc6)^
({{8{data_in[168][6]}} & 8'ha7)^
({{8{data_in[168][7]}} & 8'h65)^
({{8{data_in[169][0]}} & 8'h85)^
({{8{data_in[169][1]}} & 8'h21)^
({{8{data_in[169][2]}} & 8'h42)^
({{8{data_in[169][3]}} & 8'h84)^
({{8{data_in[169][4]}} & 8'h23)^
({{8{data_in[169][5]}} & 8'h46)^
({{8{data_in[169][6]}} & 8'h8c)^
({{8{data_in[169][7]}} & 8'h33)^
({{8{data_in[170][0]}} & 8'hf3)^
({{8{data_in[170][1]}} & 8'hcd)^
({{8{data_in[170][2]}} & 8'hb1)^
({{8{data_in[170][3]}} & 8'h49)^
({{8{data_in[170][4]}} & 8'h92)^
({{8{data_in[170][5]}} & 8'hf)^
({{8{data_in[170][6]}} & 8'h1e)^
({{8{data_in[170][7]}} & 8'h3c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[171][0]}} & 8'hbb)^
({{8{data_in[171][1]}} & 8'h5d)^
({{8{data_in[171][2]}} & 8'hba)^
({{8{data_in[171][3]}} & 8'h5f)^
({{8{data_in[171][4]}} & 8'hbe)^
({{8{data_in[171][5]}} & 8'h57)^
({{8{data_in[171][6]}} & 8'hae)^
({{8{data_in[171][7]}} & 8'h77)^
({{8{data_in[172][0]}} & 8'ha1)^
({{8{data_in[172][1]}} & 8'h69)^
({{8{data_in[172][2]}} & 8'hd2)^
({{8{data_in[172][3]}} & 8'h8f)^
({{8{data_in[172][4]}} & 8'h35)^
({{8{data_in[172][5]}} & 8'h6a)^
({{8{data_in[172][6]}} & 8'hd4)^
({{8{data_in[172][7]}} & 8'h83)^
({{8{data_in[173][0]}} & 8'h62)^
({{8{data_in[173][1]}} & 8'hc4)^
({{8{data_in[173][2]}} & 8'ha3)^
({{8{data_in[173][3]}} & 8'h6d)^
({{8{data_in[173][4]}} & 8'hda)^
({{8{data_in[173][5]}} & 8'h9f)^
({{8{data_in[173][6]}} & 8'h15)^
({{8{data_in[173][7]}} & 8'h2a)^
({{8{data_in[174][0]}} & 8'h6a)^
({{8{data_in[174][1]}} & 8'hd4)^
({{8{data_in[174][2]}} & 8'h83)^
({{8{data_in[174][3]}} & 8'h2d)^
({{8{data_in[174][4]}} & 8'h5a)^
({{8{data_in[174][5]}} & 8'hb4)^
({{8{data_in[174][6]}} & 8'h43)^
({{8{data_in[174][7]}} & 8'h86)^
({{8{data_in[175][0]}} & 8'hce)^
({{8{data_in[175][1]}} & 8'hb7)^
({{8{data_in[175][2]}} & 8'h45)^
({{8{data_in[175][3]}} & 8'h8a)^
({{8{data_in[175][4]}} & 8'h3f)^
({{8{data_in[175][5]}} & 8'h7e)^
({{8{data_in[175][6]}} & 8'hfc)^
({{8{data_in[175][7]}} & 8'hd3)^
({{8{data_in[176][0]}} & 8'h8f)^
({{8{data_in[176][1]}} & 8'h35)^
({{8{data_in[176][2]}} & 8'h6a)^
({{8{data_in[176][3]}} & 8'hd4)^
({{8{data_in[176][4]}} & 8'h83)^
({{8{data_in[176][5]}} & 8'h2d)^
({{8{data_in[176][6]}} & 8'h5a)^
({{8{data_in[176][7]}} & 8'hb4)^
({{8{data_in[177][0]}} & 8'hec)^
({{8{data_in[177][1]}} & 8'hf3)^
({{8{data_in[177][2]}} & 8'hcd)^
({{8{data_in[177][3]}} & 8'hb1)^
({{8{data_in[177][4]}} & 8'h49)^
({{8{data_in[177][5]}} & 8'h92)^
({{8{data_in[177][6]}} & 8'hf)^
({{8{data_in[177][7]}} & 8'h1e)^
({{8{data_in[178][0]}} & 8'h1c)^
({{8{data_in[178][1]}} & 8'h38)^
({{8{data_in[178][2]}} & 8'h70)^
({{8{data_in[178][3]}} & 8'he0)^
({{8{data_in[178][4]}} & 8'heb)^
({{8{data_in[178][5]}} & 8'hfd)^
({{8{data_in[178][6]}} & 8'hd1)^
({{8{data_in[178][7]}} & 8'h89)^
({{8{data_in[179][0]}} & 8'h80)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[179][1]}} & 8'h2b)^
({{8{data_in[179][2]}} & 8'h56)^
({{8{data_in[179][3]}} & 8'hac)^
({{8{data_in[179][4]}} & 8'h73)^
({{8{data_in[179][5]}} & 8'he6)^
({{8{data_in[179][6]}} & 8'he7)^
({{8{data_in[179][7]}} & 8'he5)^
({{8{data_in[180][0]}} & 8'h7e)^
({{8{data_in[180][1]}} & 8'hfc)^
({{8{data_in[180][2]}} & 8'hd3)^
({{8{data_in[180][3]}} & 8'h8d)^
({{8{data_in[180][4]}} & 8'h31)^
({{8{data_in[180][5]}} & 8'h62)^
({{8{data_in[180][6]}} & 8'hc4)^
({{8{data_in[180][7]}} & 8'ha3)^
({{8{data_in[181][0]}} & 8'ha5)^
({{8{data_in[181][1]}} & 8'h61)^
({{8{data_in[181][2]}} & 8'hc2)^
({{8{data_in[181][3]}} & 8'haf)^
({{8{data_in[181][4]}} & 8'h75)^
({{8{data_in[181][5]}} & 8'hea)^
({{8{data_in[181][6]}} & 8'hff)^
({{8{data_in[181][7]}} & 8'hd5)^
({{8{data_in[182][0]}} & 8'hef)^
({{8{data_in[182][1]}} & 8'hf5)^
({{8{data_in[182][2]}} & 8'hc1)^
({{8{data_in[182][3]}} & 8'ha9)^
({{8{data_in[182][4]}} & 8'h79)^
({{8{data_in[182][5]}} & 8'hf2)^
({{8{data_in[182][6]}} & 8'hcf)^
({{8{data_in[182][7]}} & 8'hb5)^
({{8{data_in[183][0]}} & 8'h5f)^
({{8{data_in[183][1]}} & 8'hbe)^
({{8{data_in[183][2]}} & 8'h57)^
({{8{data_in[183][3]}} & 8'hae)^
({{8{data_in[183][4]}} & 8'h77)^
({{8{data_in[183][5]}} & 8'hee)^
({{8{data_in[183][6]}} & 8'hf7)^
({{8{data_in[183][7]}} & 8'hc5)^
({{8{data_in[184][0]}} & 8'hd1)^
({{8{data_in[184][1]}} & 8'h89)^
({{8{data_in[184][2]}} & 8'h39)^
({{8{data_in[184][3]}} & 8'h72)^
({{8{data_in[184][4]}} & 8'he4)^
({{8{data_in[184][5]}} & 8'he3)^
({{8{data_in[184][6]}} & 8'hed)^
({{8{data_in[184][7]}} & 8'hf1)^
({{8{data_in[185][0]}} & 8'hdc)^
({{8{data_in[185][1]}} & 8'h93)^
({{8{data_in[185][2]}} & 8'hd)^
({{8{data_in[185][3]}} & 8'h1a)^
({{8{data_in[185][4]}} & 8'h34)^
({{8{data_in[185][5]}} & 8'h68)^
({{8{data_in[185][6]}} & 8'hd0)^
({{8{data_in[185][7]}} & 8'h8b)^
({{8{data_in[186][0]}} & 8'h24)^
({{8{data_in[186][1]}} & 8'h48)^
({{8{data_in[186][2]}} & 8'h90)^
({{8{data_in[186][3]}} & 8'hb)^
({{8{data_in[186][4]}} & 8'h16)^
({{8{data_in[186][5]}} & 8'h2c)^
({{8{data_in[186][6]}} & 8'h58)^
({{8{data_in[186][7]}} & 8'hb0)^
({{8{data_in[187][0]}} & 8'h71)^
({{8{data_in[187][1]}} & 8'he2)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[187][2]}} & 8'hef)^
({{8{data_in[187][3]}} & 8'hf5)^
({{8{data_in[187][4]}} & 8'hc1)^
({{8{data_in[187][5]}} & 8'ha9)^
({{8{data_in[187][6]}} & 8'h79)^
({{8{data_in[187][7]}} & 8'hf2)^
({{8{data_in[188][0]}} & 8'h5c)^
({{8{data_in[188][1]}} & 8'hb8)^
({{8{data_in[188][2]}} & 8'h5b)^
({{8{data_in[188][3]}} & 8'hb6)^
({{8{data_in[188][4]}} & 8'h47)^
({{8{data_in[188][5]}} & 8'h8e)^
({{8{data_in[188][6]}} & 8'h37)^
({{8{data_in[188][7]}} & 8'h6e)^
({{8{data_in[189][0]}} & 8'h51)^
({{8{data_in[189][1]}} & 8'ha2)^
({{8{data_in[189][2]}} & 8'h6f)^
({{8{data_in[189][3]}} & 8'hde)^
({{8{data_in[189][4]}} & 8'h97)^
({{8{data_in[189][5]}} & 8'h5)^
({{8{data_in[189][6]}} & 8'ha)^
({{8{data_in[189][7]}} & 8'h14)^
({{8{data_in[190][0]}} & 8'h3d)^
({{8{data_in[190][1]}} & 8'h7a)^
({{8{data_in[190][2]}} & 8'hf4)^
({{8{data_in[190][3]}} & 8'hc3)^
({{8{data_in[190][4]}} & 8'had)^
({{8{data_in[190][5]}} & 8'h71)^
({{8{data_in[190][6]}} & 8'he2)^
({{8{data_in[190][7]}} & 8'hef)^
({{8{data_in[191][0]}} & 8'hb8)^
({{8{data_in[191][1]}} & 8'h5b)^
({{8{data_in[191][2]}} & 8'hb6)^
({{8{data_in[191][3]}} & 8'h47)^
({{8{data_in[191][4]}} & 8'h8e)^
({{8{data_in[191][5]}} & 8'h37)^
({{8{data_in[191][6]}} & 8'h6e)^
({{8{data_in[191][7]}} & 8'hdc)^
({{8{data_in[192][0]}} & 8'h38)^
({{8{data_in[192][1]}} & 8'h70)^
({{8{data_in[192][2]}} & 8'he0)^
({{8{data_in[192][3]}} & 8'heb)^
({{8{data_in[192][4]}} & 8'hfd)^
({{8{data_in[192][5]}} & 8'hd1)^
({{8{data_in[192][6]}} & 8'h89)^
({{8{data_in[192][7]}} & 8'h39)^
({{8{data_in[193][0]}} & 8'h19)^
({{8{data_in[193][1]}} & 8'h32)^
({{8{data_in[193][2]}} & 8'h64)^
({{8{data_in[193][3]}} & 8'hc8)^
({{8{data_in[193][4]}} & 8'hbb)^
({{8{data_in[193][5]}} & 8'h5d)^
({{8{data_in[193][6]}} & 8'hba)^
({{8{data_in[193][7]}} & 8'h5f)^
({{8{data_in[194][0]}} & 8'h6c)^
({{8{data_in[194][1]}} & 8'hd8)^
({{8{data_in[194][2]}} & 8'h9b)^
({{8{data_in[194][3]}} & 8'h1d)^
({{8{data_in[194][4]}} & 8'h3a)^
({{8{data_in[194][5]}} & 8'h74)^
({{8{data_in[194][6]}} & 8'he8)^
({{8{data_in[194][7]}} & 8'hfb)^
({{8{data_in[195][0]}} & 8'h2)^
({{8{data_in[195][1]}} & 8'h4)^
({{8{data_in[195][2]}} & 8'h8)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[195][3]}} & 8'h10)^
({{8{data_in[195][4]}} & 8'h20)^
({{8{data_in[195][5]}} & 8'h40)^
({{8{data_in[195][6]}} & 8'h80)^
({{8{data_in[195][7]}} & 8'h2b)^
({{8{data_in[196][0]}} & 8'h9)^
({{8{data_in[196][1]}} & 8'h12)^
({{8{data_in[196][2]}} & 8'h24)^
({{8{data_in[196][3]}} & 8'h48)^
({{8{data_in[196][4]}} & 8'h90)^
({{8{data_in[196][5]}} & 8'hb)^
({{8{data_in[196][6]}} & 8'h16)^
({{8{data_in[196][7]}} & 8'h2c)^
({{8{data_in[197][0]}} & 8'h50)^
({{8{data_in[197][1]}} & 8'ha0)^
({{8{data_in[197][2]}} & 8'h6b)^
({{8{data_in[197][3]}} & 8'hd6)^
({{8{data_in[197][4]}} & 8'h87)^
({{8{data_in[197][5]}} & 8'h25)^
({{8{data_in[197][6]}} & 8'h4a)^
({{8{data_in[197][7]}} & 8'h94)^
({{8{data_in[198][0]}} & 8'hc4)^
({{8{data_in[198][1]}} & 8'ha3)^
({{8{data_in[198][2]}} & 8'h6d)^
({{8{data_in[198][3]}} & 8'hda)^
({{8{data_in[198][4]}} & 8'h9f)^
({{8{data_in[198][5]}} & 8'h15)^
({{8{data_in[198][6]}} & 8'h2a)^
({{8{data_in[198][7]}} & 8'h54)^
({{8{data_in[199][0]}} & 8'hac)^
({{8{data_in[199][1]}} & 8'h73)^
({{8{data_in[199][2]}} & 8'he6)^
({{8{data_in[199][3]}} & 8'he7)^
({{8{data_in[199][4]}} & 8'he5)^
({{8{data_in[199][5]}} & 8'he1)^
({{8{data_in[199][6]}} & 8'he9)^
({{8{data_in[199][7]}} & 8'hf9)^
({{8{data_in[200][0]}} & 8'h25)^
({{8{data_in[200][1]}} & 8'h4a)^
({{8{data_in[200][2]}} & 8'h94)^
({{8{data_in[200][3]}} & 8'h3)^
({{8{data_in[200][4]}} & 8'h6)^
({{8{data_in[200][5]}} & 8'hc)^
({{8{data_in[200][6]}} & 8'h18)^
({{8{data_in[200][7]}} & 8'h30)^
({{8{data_in[201][0]}} & 8'he2)^
({{8{data_in[201][1]}} & 8'hef)^
({{8{data_in[201][2]}} & 8'hf5)^
({{8{data_in[201][3]}} & 8'hc1)^
({{8{data_in[201][4]}} & 8'ha9)^
({{8{data_in[201][5]}} & 8'h79)^
({{8{data_in[201][6]}} & 8'hf2)^
({{8{data_in[201][7]}} & 8'hcf)^
({{8{data_in[202][0]}} & 8'h5c)^
({{8{data_in[202][1]}} & 8'hb8)^
({{8{data_in[202][2]}} & 8'h5b)^
({{8{data_in[202][3]}} & 8'hb6)^
({{8{data_in[202][4]}} & 8'h47)^
({{8{data_in[202][5]}} & 8'h8e)^
({{8{data_in[202][6]}} & 8'h37)^
({{8{data_in[202][7]}} & 8'h6e)^
({{8{data_in[203][0]}} & 8'ha1)^
({{8{data_in[203][1]}} & 8'h69)^
({{8{data_in[203][2]}} & 8'hd2)^
({{8{data_in[203][3]}} & 8'h8f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[203][4]}} & 8'h35)^
({{8{data_in[203][5]}} & 8'h6a)^
({{8{data_in[203][6]}} & 8'hd4)^
({{8{data_in[203][7]}} & 8'h83)^
({{8{data_in[204][0]}} & 8'h50)^
({{8{data_in[204][1]}} & 8'ha0)^
({{8{data_in[204][2]}} & 8'h6b)^
({{8{data_in[204][3]}} & 8'hd6)^
({{8{data_in[204][4]}} & 8'h87)^
({{8{data_in[204][5]}} & 8'h25)^
({{8{data_in[204][6]}} & 8'h4a)^
({{8{data_in[204][7]}} & 8'h94)^
({{8{data_in[205][0]}} & 8'hba)^
({{8{data_in[205][1]}} & 8'h5f)^
({{8{data_in[205][2]}} & 8'hbe)^
({{8{data_in[205][3]}} & 8'h57)^
({{8{data_in[205][4]}} & 8'hae)^
({{8{data_in[205][5]}} & 8'h77)^
({{8{data_in[205][6]}} & 8'hee)^
({{8{data_in[205][7]}} & 8'hf7)^
({{8{data_in[206][0]}} & 8'h97)^
({{8{data_in[206][1]}} & 8'h5)^
({{8{data_in[206][2]}} & 8'ha)^
({{8{data_in[206][3]}} & 8'h14)^
({{8{data_in[206][4]}} & 8'h28)^
({{8{data_in[206][5]}} & 8'h50)^
({{8{data_in[206][6]}} & 8'ha0)^
({{8{data_in[206][7]}} & 8'h6b)^
({{8{data_in[207][0]}} & 8'h4f)^
({{8{data_in[207][1]}} & 8'h9e)^
({{8{data_in[207][2]}} & 8'h17)^
({{8{data_in[207][3]}} & 8'h2e)^
({{8{data_in[207][4]}} & 8'h5c)^
({{8{data_in[207][5]}} & 8'hb8)^
({{8{data_in[207][6]}} & 8'h5b)^
({{8{data_in[207][7]}} & 8'hb6)^
({{8{data_in[208][0]}} & 8'h13)^
({{8{data_in[208][1]}} & 8'h26)^
({{8{data_in[208][2]}} & 8'h4c)^
({{8{data_in[208][3]}} & 8'h98)^
({{8{data_in[208][4]}} & 8'h1b)^
({{8{data_in[208][5]}} & 8'h36)^
({{8{data_in[208][6]}} & 8'h6c)^
({{8{data_in[208][7]}} & 8'hd8)^
({{8{data_in[209][0]}} & 8'hf6)^
({{8{data_in[209][1]}} & 8'hc7)^
({{8{data_in[209][2]}} & 8'ha5)^
({{8{data_in[209][3]}} & 8'h61)^
({{8{data_in[209][4]}} & 8'hc2)^
({{8{data_in[209][5]}} & 8'haf)^
({{8{data_in[209][6]}} & 8'h75)^
({{8{data_in[209][7]}} & 8'hea)^
({{8{data_in[210][0]}} & 8'hd8)^
({{8{data_in[210][1]}} & 8'h9b)^
({{8{data_in[210][2]}} & 8'h1d)^
({{8{data_in[210][3]}} & 8'h3a)^
({{8{data_in[210][4]}} & 8'h74)^
({{8{data_in[210][5]}} & 8'he8)^
({{8{data_in[210][6]}} & 8'hfb)^
({{8{data_in[210][7]}} & 8'hdd)^
({{8{data_in[211][0]}} & 8'hd)^
({{8{data_in[211][1]}} & 8'h1a)^
({{8{data_in[211][2]}} & 8'h34)^
({{8{data_in[211][3]}} & 8'h68)^
({{8{data_in[211][4]}} & 8'hd0)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[211][5]}} & 8'h8b)^
({{8{data_in[211][6]}} & 8'h3d)^
({{8{data_in[211][7]}} & 8'h7a)^
({{8{data_in[212][0]}} & 8'hdb)^
({{8{data_in[212][1]}} & 8'h9d)^
({{8{data_in[212][2]}} & 8'h11)^
({{8{data_in[212][3]}} & 8'h22)^
({{8{data_in[212][4]}} & 8'h44)^
({{8{data_in[212][5]}} & 8'h88)^
({{8{data_in[212][6]}} & 8'h3b)^
({{8{data_in[212][7]}} & 8'h76)^
({{8{data_in[213][0]}} & 8'h4d)^
({{8{data_in[213][1]}} & 8'h9a)^
({{8{data_in[213][2]}} & 8'h1f)^
({{8{data_in[213][3]}} & 8'h3e)^
({{8{data_in[213][4]}} & 8'h7c)^
({{8{data_in[213][5]}} & 8'hf8)^
({{8{data_in[213][6]}} & 8'hdb)^
({{8{data_in[213][7]}} & 8'h9d)^
({{8{data_in[214][0]}} & 8'he3)^
({{8{data_in[214][1]}} & 8'hd)^
({{8{data_in[214][2]}} & 8'hf1)^
({{8{data_in[214][3]}} & 8'hc9)^
({{8{data_in[214][4]}} & 8'hb9)^
({{8{data_in[214][5]}} & 8'h59)^
({{8{data_in[214][6]}} & 8'hb2)^
({{8{data_in[214][7]}} & 8'h4f)^
({{8{data_in[215][0]}} & 8'h6b)^
({{8{data_in[215][1]}} & 8'hd6)^
({{8{data_in[215][2]}} & 8'h87)^
({{8{data_in[215][3]}} & 8'h25)^
({{8{data_in[215][4]}} & 8'h4a)^
({{8{data_in[215][5]}} & 8'h94)^
({{8{data_in[215][6]}} & 8'h3)^
({{8{data_in[215][7]}} & 8'h6)^
({{8{data_in[216][0]}} & 8'h45)^
({{8{data_in[216][1]}} & 8'h8a)^
({{8{data_in[216][2]}} & 8'h3f)^
({{8{data_in[216][3]}} & 8'h7e)^
({{8{data_in[216][4]}} & 8'hfc)^
({{8{data_in[216][5]}} & 8'hd3)^
({{8{data_in[216][6]}} & 8'h8d)^
({{8{data_in[216][7]}} & 8'h31)^
({{8{data_in[217][0]}} & 8'hfa)^
({{8{data_in[217][1]}} & 8'hdf)^
({{8{data_in[217][2]}} & 8'h95)^
({{8{data_in[217][3]}} & 8'h1)^
({{8{data_in[217][4]}} & 8'h2)^
({{8{data_in[217][5]}} & 8'h4)^
({{8{data_in[217][6]}} & 8'h8)^
({{8{data_in[217][7]}} & 8'h10)^
({{8{data_in[218][0]}} & 8'h33)^
({{8{data_in[218][1]}} & 8'h66)^
({{8{data_in[218][2]}} & 8'hcc)^
({{8{data_in[218][3]}} & 8'hb3)^
({{8{data_in[218][4]}} & 8'h4d)^
({{8{data_in[218][5]}} & 8'h9a)^
({{8{data_in[218][6]}} & 8'h1f)^
({{8{data_in[218][7]}} & 8'h3e)^
({{8{data_in[219][0]}} & 8'ha0)^
({{8{data_in[219][1]}} & 8'h6b)^
({{8{data_in[219][2]}} & 8'hd6)^
({{8{data_in[219][3]}} & 8'h87)^
({{8{data_in[219][4]}} & 8'h25)^
({{8{data_in[219][5]}} & 8'h4a)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[219][6]}} & 8'h94)^
({{8{data_in[219][7]}} & 8'h3)^
({{8{data_in[220][0]}} & 8'h2c)^
({{8{data_in[220][1]}} & 8'h58)^
({{8{data_in[220][2]}} & 8'hb0)^
({{8{data_in[220][3]}} & 8'h4b)^
({{8{data_in[220][4]}} & 8'h96)^
({{8{data_in[220][5]}} & 8'h7)^
({{8{data_in[220][6]}} & 8'he)^
({{8{data_in[220][7]}} & 8'h1c)^
({{8{data_in[221][0]}} & 8'h4e)^
({{8{data_in[221][1]}} & 8'h9c)^
({{8{data_in[221][2]}} & 8'h13)^
({{8{data_in[221][3]}} & 8'h26)^
({{8{data_in[221][4]}} & 8'h4c)^
({{8{data_in[221][5]}} & 8'h98)^
({{8{data_in[221][6]}} & 8'h1b)^
({{8{data_in[221][7]}} & 8'h36)^
({{8{data_in[222][0]}} & 8'hbd)^
({{8{data_in[222][1]}} & 8'h51)^
({{8{data_in[222][2]}} & 8'ha2)^
({{8{data_in[222][3]}} & 8'h6f)^
({{8{data_in[222][4]}} & 8'hde)^
({{8{data_in[222][5]}} & 8'h97)^
({{8{data_in[222][6]}} & 8'h5)^
({{8{data_in[222][7]}} & 8'ha)^
({{8{data_in[223][0]}} & 8'h74)^
({{8{data_in[223][1]}} & 8'he8)^
({{8{data_in[223][2]}} & 8'hfb)^
({{8{data_in[223][3]}} & 8'hdd)^
({{8{data_in[223][4]}} & 8'h91)^
({{8{data_in[223][5]}} & 8'h9)^
({{8{data_in[223][6]}} & 8'h12)^
({{8{data_in[223][7]}} & 8'h24)^
({{8{data_in[224][0]}} & 8'hd5)^
({{8{data_in[224][1]}} & 8'h81)^
({{8{data_in[224][2]}} & 8'h29)^
({{8{data_in[224][3]}} & 8'h52)^
({{8{data_in[224][4]}} & 8'ha4)^
({{8{data_in[224][5]}} & 8'h63)^
({{8{data_in[224][6]}} & 8'hc6)^
({{8{data_in[224][7]}} & 8'ha7)^
({{8{data_in[225][0]}} & 8'h7c)^
({{8{data_in[225][1]}} & 8'hf8)^
({{8{data_in[225][2]}} & 8'hdb)^
({{8{data_in[225][3]}} & 8'h9d)^
({{8{data_in[225][4]}} & 8'h11)^
({{8{data_in[225][5]}} & 8'h22)^
({{8{data_in[225][6]}} & 8'h44)^
({{8{data_in[225][7]}} & 8'h88)^
({{8{data_in[226][0]}} & 8'ha3)^
({{8{data_in[226][1]}} & 8'h6d)^
({{8{data_in[226][2]}} & 8'hda)^
({{8{data_in[226][3]}} & 8'h9f)^
({{8{data_in[226][4]}} & 8'h15)^
({{8{data_in[226][5]}} & 8'h2a)^
({{8{data_in[226][6]}} & 8'h54)^
({{8{data_in[226][7]}} & 8'ha8)^
({{8{data_in[227][0]}} & 8'hf5)^
({{8{data_in[227][1]}} & 8'hc1)^
({{8{data_in[227][2]}} & 8'ha9)^
({{8{data_in[227][3]}} & 8'h79)^
({{8{data_in[227][4]}} & 8'hf2)^
({{8{data_in[227][5]}} & 8'hcf)^
({{8{data_in[227][6]}} & 8'hb5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[227][7]}} & 8'h41)^
({{8{data_in[228][0]}} & 8'h4a)^
({{8{data_in[228][1]}} & 8'h94)^
({{8{data_in[228][2]}} & 8'h3)^
({{8{data_in[228][3]}} & 8'h6)^
({{8{data_in[228][4]}} & 8'hc)^
({{8{data_in[228][5]}} & 8'h18)^
({{8{data_in[228][6]}} & 8'h30)^
({{8{data_in[228][7]}} & 8'h60)^
({{8{data_in[229][0]}} & 8'hc0)^
({{8{data_in[229][1]}} & 8'hab)^
({{8{data_in[229][2]}} & 8'h7d)^
({{8{data_in[229][3]}} & 8'hfa)^
({{8{data_in[229][4]}} & 8'hdf)^
({{8{data_in[229][5]}} & 8'h95)^
({{8{data_in[229][6]}} & 8'h1)^
({{8{data_in[229][7]}} & 8'h2)^
({{8{data_in[230][0]}} & 8'h73)^
({{8{data_in[230][1]}} & 8'he6)^
({{8{data_in[230][2]}} & 8'he7)^
({{8{data_in[230][3]}} & 8'he5)^
({{8{data_in[230][4]}} & 8'he1)^
({{8{data_in[230][5]}} & 8'he9)^
({{8{data_in[230][6]}} & 8'hf9)^
({{8{data_in[230][7]}} & 8'hd9)^
({{8{data_in[231][0]}} & 8'h6d)^
({{8{data_in[231][1]}} & 8'hda)^
({{8{data_in[231][2]}} & 8'h9f)^
({{8{data_in[231][3]}} & 8'h15)^
({{8{data_in[231][4]}} & 8'h2a)^
({{8{data_in[231][5]}} & 8'h54)^
({{8{data_in[231][6]}} & 8'ha8)^
({{8{data_in[231][7]}} & 8'h7b)^
({{8{data_in[232][0]}} & 8'h5b)^
({{8{data_in[232][1]}} & 8'hb6)^
({{8{data_in[232][2]}} & 8'h47)^
({{8{data_in[232][3]}} & 8'h8e)^
({{8{data_in[232][4]}} & 8'h37)^
({{8{data_in[232][5]}} & 8'h6e)^
({{8{data_in[232][6]}} & 8'hdc)^
({{8{data_in[232][7]}} & 8'h93)^
({{8{data_in[233][0]}} & 8'hf6)^
({{8{data_in[233][1]}} & 8'hc7)^
({{8{data_in[233][2]}} & 8'ha5)^
({{8{data_in[233][3]}} & 8'h61)^
({{8{data_in[233][4]}} & 8'hc2)^
({{8{data_in[233][5]}} & 8'haf)^
({{8{data_in[233][6]}} & 8'h75)^
({{8{data_in[233][7]}} & 8'hea)^
({{8{data_in[234][0]}} & 8'h7d)^
({{8{data_in[234][1]}} & 8'hfa)^
({{8{data_in[234][2]}} & 8'hdf)^
({{8{data_in[234][3]}} & 8'h95)^
({{8{data_in[234][4]}} & 8'h1)^
({{8{data_in[234][5]}} & 8'h2)^
({{8{data_in[234][6]}} & 8'h4)^
({{8{data_in[234][7]}} & 8'h8)^
({{8{data_in[235][0]}} & 8'hf6)^
({{8{data_in[235][1]}} & 8'hc7)^
({{8{data_in[235][2]}} & 8'ha5)^
({{8{data_in[235][3]}} & 8'h61)^
({{8{data_in[235][4]}} & 8'hc2)^
({{8{data_in[235][5]}} & 8'haf)^
({{8{data_in[235][6]}} & 8'h75)^
({{8{data_in[235][7]}} & 8'hea)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[236][0]}} & 8'hf8)^
({{8{data_in[236][1]}} & 8'hdb)^
({{8{data_in[236][2]}} & 8'h9d)^
({{8{data_in[236][3]}} & 8'h11)^
({{8{data_in[236][4]}} & 8'h22)^
({{8{data_in[236][5]}} & 8'h44)^
({{8{data_in[236][6]}} & 8'h88)^
({{8{data_in[236][7]}} & 8'h3b)^
({{8{data_in[237][0]}} & 8'h27)^
({{8{data_in[237][1]}} & 8'h4e)^
({{8{data_in[237][2]}} & 8'h9c)^
({{8{data_in[237][3]}} & 8'h13)^
({{8{data_in[237][4]}} & 8'h26)^
({{8{data_in[237][5]}} & 8'h4c)^
({{8{data_in[237][6]}} & 8'h98)^
({{8{data_in[237][7]}} & 8'h1b)^
({{8{data_in[238][0]}} & 8'h22)^
({{8{data_in[238][1]}} & 8'h44)^
({{8{data_in[238][2]}} & 8'h88)^
({{8{data_in[238][3]}} & 8'h3b)^
({{8{data_in[238][4]}} & 8'h76)^
({{8{data_in[238][5]}} & 8'hec)^
({{8{data_in[238][6]}} & 8'hf3)^
({{8{data_in[238][7]}} & 8'hcd)^
({{8{data_in[239][0]}} & 8'hc8)^
({{8{data_in[239][1]}} & 8'hbb)^
({{8{data_in[239][2]}} & 8'h5d)^
({{8{data_in[239][3]}} & 8'hba)^
({{8{data_in[239][4]}} & 8'h5f)^
({{8{data_in[239][5]}} & 8'hbe)^
({{8{data_in[239][6]}} & 8'h57)^
({{8{data_in[239][7]}} & 8'hae)^
({{8{data_in[240][0]}} & 8'h1)^
({{8{data_in[240][1]}} & 8'h2)^
({{8{data_in[240][2]}} & 8'h4)^
({{8{data_in[240][3]}} & 8'h8)^
({{8{data_in[240][4]}} & 8'h10)^
({{8{data_in[240][5]}} & 8'h20)^
({{8{data_in[240][6]}} & 8'h40)^
({{8{data_in[240][7]}} & 8'h80)^
({{8{data_in[241][0]}} & 8'h4d)^
({{8{data_in[241][1]}} & 8'h9a)^
({{8{data_in[241][2]}} & 8'h1f)^
({{8{data_in[241][3]}} & 8'h3e)^
({{8{data_in[241][4]}} & 8'h7c)^
({{8{data_in[241][5]}} & 8'hf8)^
({{8{data_in[241][6]}} & 8'hdb)^
({{8{data_in[241][7]}} & 8'h9d);

data_out[242] = ({8{data_in[0][0]}} & 8'hb)^
({8{data_in[0][1]}} & 8'h16)^
({8{data_in[0][2]}} & 8'h2c)^
({8{data_in[0][3]}} & 8'h58)^
({8{data_in[0][4]}} & 8'hb0)^
({8{data_in[0][5]}} & 8'h4b)^
({8{data_in[0][6]}} & 8'h96)^
({8{data_in[0][7]}} & 8'h7)^
({8{data_in[1][0]}} & 8'h6b)^
({8{data_in[1][1]}} & 8'hd6)^
({8{data_in[1][2]}} & 8'h87)^
({8{data_in[1][3]}} & 8'h25)^
({8{data_in[1][4]}} & 8'h4a)^
({8{data_in[1][5]}} & 8'h94)^
({8{data_in[1][6]}} & 8'h3)^
({8{data_in[1][7]}} & 8'h6)^
({8{data_in[2][0]}} & 8'ha3)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[2][1]}} & 8'h6d)^
({{8{data_in[2][2]}} & 8'hda)^
({{8{data_in[2][3]}} & 8'h9f)^
({{8{data_in[2][4]}} & 8'h15)^
({{8{data_in[2][5]}} & 8'h2a)^
({{8{data_in[2][6]}} & 8'h54)^
({{8{data_in[2][7]}} & 8'ha8)^
({{8{data_in[3][0]}} & 8'h65)^
({{8{data_in[3][1]}} & 8'hca)^
({{8{data_in[3][2]}} & 8'hbf)^
({{8{data_in[3][3]}} & 8'h55)^
({{8{data_in[3][4]}} & 8'haa)^
({{8{data_in[3][5]}} & 8'h7f)^
({{8{data_in[3][6]}} & 8'hfe)^
({{8{data_in[3][7]}} & 8'hd7)^
({{8{data_in[4][0]}} & 8'hdf)^
({{8{data_in[4][1]}} & 8'h95)^
({{8{data_in[4][2]}} & 8'h1)^
({{8{data_in[4][3]}} & 8'h2)^
({{8{data_in[4][4]}} & 8'h4)^
({{8{data_in[4][5]}} & 8'h8)^
({{8{data_in[4][6]}} & 8'h10)^
({{8{data_in[4][7]}} & 8'h20)^
({{8{data_in[5][0]}} & 8'hec)^
({{8{data_in[5][1]}} & 8'hf3)^
({{8{data_in[5][2]}} & 8'hcd)^
({{8{data_in[5][3]}} & 8'hb1)^
({{8{data_in[5][4]}} & 8'h49)^
({{8{data_in[5][5]}} & 8'h92)^
({{8{data_in[5][6]}} & 8'hf)^
({{8{data_in[5][7]}} & 8'h1e)^
({{8{data_in[6][0]}} & 8'h18)^
({{8{data_in[6][1]}} & 8'h30)^
({{8{data_in[6][2]}} & 8'h60)^
({{8{data_in[6][3]}} & 8'hc0)^
({{8{data_in[6][4]}} & 8'hab)^
({{8{data_in[6][5]}} & 8'h7d)^
({{8{data_in[6][6]}} & 8'hfa)^
({{8{data_in[6][7]}} & 8'hdf)^
({{8{data_in[7][0]}} & 8'h49)^
({{8{data_in[7][1]}} & 8'h92)^
({{8{data_in[7][2]}} & 8'hf)^
({{8{data_in[7][3]}} & 8'h1e)^
({{8{data_in[7][4]}} & 8'h3c)^
({{8{data_in[7][5]}} & 8'h78)^
({{8{data_in[7][6]}} & 8'hf0)^
({{8{data_in[7][7]}} & 8'hcb)^
({{8{data_in[8][0]}} & 8'hd8)^
({{8{data_in[8][1]}} & 8'h9b)^
({{8{data_in[8][2]}} & 8'h1d)^
({{8{data_in[8][3]}} & 8'h3a)^
({{8{data_in[8][4]}} & 8'h74)^
({{8{data_in[8][5]}} & 8'he8)^
({{8{data_in[8][6]}} & 8'hfb)^
({{8{data_in[8][7]}} & 8'hdd)^
({{8{data_in[9][0]}} & 8'h3f)^
({{8{data_in[9][1]}} & 8'h7e)^
({{8{data_in[9][2]}} & 8'hfc)^
({{8{data_in[9][3]}} & 8'hd3)^
({{8{data_in[9][4]}} & 8'h8d)^
({{8{data_in[9][5]}} & 8'h31)^
({{8{data_in[9][6]}} & 8'h62)^
({{8{data_in[9][7]}} & 8'hc4)^
({{8{data_in[10][0]}} & 8'hea)^
({{8{data_in[10][1]}} & 8'hff)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[10][2]}} & 8'hd5)^
({{8{data_in[10][3]}} & 8'h81)^
({{8{data_in[10][4]}} & 8'h29)^
({{8{data_in[10][5]}} & 8'h52)^
({{8{data_in[10][6]}} & 8'ha4)^
({{8{data_in[10][7]}} & 8'h63)^
({{8{data_in[11][0]}} & 8'h87)^
({{8{data_in[11][1]}} & 8'h25)^
({{8{data_in[11][2]}} & 8'h4a)^
({{8{data_in[11][3]}} & 8'h94)^
({{8{data_in[11][4]}} & 8'h3)^
({{8{data_in[11][5]}} & 8'h6)^
({{8{data_in[11][6]}} & 8'hc)^
({{8{data_in[11][7]}} & 8'h18)^
({{8{data_in[12][0]}} & 8'h32)^
({{8{data_in[12][1]}} & 8'h64)^
({{8{data_in[12][2]}} & 8'hc8)^
({{8{data_in[12][3]}} & 8'hbb)^
({{8{data_in[12][4]}} & 8'h5d)^
({{8{data_in[12][5]}} & 8'hba)^
({{8{data_in[12][6]}} & 8'h5f)^
({{8{data_in[12][7]}} & 8'hbe)^
({{8{data_in[13][0]}} & 8'hd6)^
({{8{data_in[13][1]}} & 8'h87)^
({{8{data_in[13][2]}} & 8'h25)^
({{8{data_in[13][3]}} & 8'h4a)^
({{8{data_in[13][4]}} & 8'h94)^
({{8{data_in[13][5]}} & 8'h3)^
({{8{data_in[13][6]}} & 8'h6)^
({{8{data_in[13][7]}} & 8'hc)^
({{8{data_in[14][0]}} & 8'hf4)^
({{8{data_in[14][1]}} & 8'hc3)^
({{8{data_in[14][2]}} & 8'had)^
({{8{data_in[14][3]}} & 8'h71)^
({{8{data_in[14][4]}} & 8'he2)^
({{8{data_in[14][5]}} & 8'hef)^
({{8{data_in[14][6]}} & 8'hf5)^
({{8{data_in[14][7]}} & 8'hc1)^
({{8{data_in[15][0]}} & 8'h9e)^
({{8{data_in[15][1]}} & 8'h17)^
({{8{data_in[15][2]}} & 8'h2e)^
({{8{data_in[15][3]}} & 8'h5c)^
({{8{data_in[15][4]}} & 8'hb8)^
({{8{data_in[15][5]}} & 8'h5b)^
({{8{data_in[15][6]}} & 8'hb6)^
({{8{data_in[15][7]}} & 8'h47)^
({{8{data_in[16][0]}} & 8'ha4)^
({{8{data_in[16][1]}} & 8'h63)^
({{8{data_in[16][2]}} & 8'hc6)^
({{8{data_in[16][3]}} & 8'ha7)^
({{8{data_in[16][4]}} & 8'h65)^
({{8{data_in[16][5]}} & 8'hca)^
({{8{data_in[16][6]}} & 8'hbf)^
({{8{data_in[16][7]}} & 8'h55)^
({{8{data_in[17][0]}} & 8'h4a)^
({{8{data_in[17][1]}} & 8'h94)^
({{8{data_in[17][2]}} & 8'h3)^
({{8{data_in[17][3]}} & 8'h6)^
({{8{data_in[17][4]}} & 8'hc)^
({{8{data_in[17][5]}} & 8'h18)^
({{8{data_in[17][6]}} & 8'h30)^
({{8{data_in[17][7]}} & 8'h60)^
({{8{data_in[18][0]}} & 8'hc7)^
({{8{data_in[18][1]}} & 8'ha5)^
({{8{data_in[18][2]}} & 8'h61})^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[18][3]}} & 8'hc2)^
({{8{data_in[18][4]}} & 8'haf)^
({{8{data_in[18][5]}} & 8'h75)^
({{8{data_in[18][6]}} & 8'hea)^
({{8{data_in[18][7]}} & 8'hff)^
({{8{data_in[19][0]}} & 8'ha7)^
({{8{data_in[19][1]}} & 8'h65)^
({{8{data_in[19][2]}} & 8'hca)^
({{8{data_in[19][3]}} & 8'hbf)^
({{8{data_in[19][4]}} & 8'h55)^
({{8{data_in[19][5]}} & 8'haa)^
({{8{data_in[19][6]}} & 8'h7f)^
({{8{data_in[19][7]}} & 8'hfe)^
({{8{data_in[20][0]}} & 8'h1b)^
({{8{data_in[20][1]}} & 8'h36)^
({{8{data_in[20][2]}} & 8'h6c)^
({{8{data_in[20][3]}} & 8'hd8)^
({{8{data_in[20][4]}} & 8'h9b)^
({{8{data_in[20][5]}} & 8'h1d)^
({{8{data_in[20][6]}} & 8'h3a)^
({{8{data_in[20][7]}} & 8'h74)^
({{8{data_in[21][0]}} & 8'he8)^
({{8{data_in[21][1]}} & 8'hfb)^
({{8{data_in[21][2]}} & 8'hdd)^
({{8{data_in[21][3]}} & 8'h91)^
({{8{data_in[21][4]}} & 8'h9)^
({{8{data_in[21][5]}} & 8'h12)^
({{8{data_in[21][6]}} & 8'h24)^
({{8{data_in[21][7]}} & 8'h48)^
({{8{data_in[22][0]}} & 8'h37)^
({{8{data_in[22][1]}} & 8'h6e)^
({{8{data_in[22][2]}} & 8'hdc)^
({{8{data_in[22][3]}} & 8'h93)^
({{8{data_in[22][4]}} & 8'hd)^
({{8{data_in[22][5]}} & 8'h1a)^
({{8{data_in[22][6]}} & 8'h34)^
({{8{data_in[22][7]}} & 8'h68)^
({{8{data_in[23][0]}} & 8'h22)^
({{8{data_in[23][1]}} & 8'h44)^
({{8{data_in[23][2]}} & 8'h88)^
({{8{data_in[23][3]}} & 8'h3b)^
({{8{data_in[23][4]}} & 8'h76)^
({{8{data_in[23][5]}} & 8'hec)^
({{8{data_in[23][6]}} & 8'hf3)^
({{8{data_in[23][7]}} & 8'hcd)^
({{8{data_in[24][0]}} & 8'hca)^
({{8{data_in[24][1]}} & 8'hbf)^
({{8{data_in[24][2]}} & 8'h55)^
({{8{data_in[24][3]}} & 8'haa)^
({{8{data_in[24][4]}} & 8'h7f)^
({{8{data_in[24][5]}} & 8'hfe)^
({{8{data_in[24][6]}} & 8'hd7)^
({{8{data_in[24][7]}} & 8'h85)^
({{8{data_in[25][0]}} & 8'h24)^
({{8{data_in[25][1]}} & 8'h48)^
({{8{data_in[25][2]}} & 8'h90)^
({{8{data_in[25][3]}} & 8'hb)^
({{8{data_in[25][4]}} & 8'h16)^
({{8{data_in[25][5]}} & 8'h2c)^
({{8{data_in[25][6]}} & 8'h58)^
({{8{data_in[25][7]}} & 8'hb0)^
({{8{data_in[26][0]}} & 8'h9c)^
({{8{data_in[26][1]}} & 8'h13)^
({{8{data_in[26][2]}} & 8'h26)^
({{8{data_in[26][3]}} & 8'h4c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[26][4]}} & 8'h98)^
({{8{data_in[26][5]}} & 8'h1b)^
({{8{data_in[26][6]}} & 8'h36)^
({{8{data_in[26][7]}} & 8'h6c)^
({{8{data_in[27][0]}} & 8'h68)^
({{8{data_in[27][1]}} & 8'hd0)^
({{8{data_in[27][2]}} & 8'h8b)^
({{8{data_in[27][3]}} & 8'h3d)^
({{8{data_in[27][4]}} & 8'h7a)^
({{8{data_in[27][5]}} & 8'hf4)^
({{8{data_in[27][6]}} & 8'hc3)^
({{8{data_in[27][7]}} & 8'had)^
({{8{data_in[28][0]}} & 8'hf5)^
({{8{data_in[28][1]}} & 8'hc1)^
({{8{data_in[28][2]}} & 8'ha9)^
({{8{data_in[28][3]}} & 8'h79)^
({{8{data_in[28][4]}} & 8'hf2)^
({{8{data_in[28][5]}} & 8'hcf)^
({{8{data_in[28][6]}} & 8'hb5)^
({{8{data_in[28][7]}} & 8'h41)^
({{8{data_in[29][0]}} & 8'hd9)^
({{8{data_in[29][1]}} & 8'h99)^
({{8{data_in[29][2]}} & 8'h19)^
({{8{data_in[29][3]}} & 8'h32)^
({{8{data_in[29][4]}} & 8'h64)^
({{8{data_in[29][5]}} & 8'hc8)^
({{8{data_in[29][6]}} & 8'hbb)^
({{8{data_in[29][7]}} & 8'h5d)^
({{8{data_in[30][0]}} & 8'hf0)^
({{8{data_in[30][1]}} & 8'hcb)^
({{8{data_in[30][2]}} & 8'hbd)^
({{8{data_in[30][3]}} & 8'h51)^
({{8{data_in[30][4]}} & 8'ha2)^
({{8{data_in[30][5]}} & 8'h6f)^
({{8{data_in[30][6]}} & 8'hde)^
({{8{data_in[30][7]}} & 8'h97)^
({{8{data_in[31][0]}} & 8'hff)^
({{8{data_in[31][1]}} & 8'hd5)^
({{8{data_in[31][2]}} & 8'h81)^
({{8{data_in[31][3]}} & 8'h29)^
({{8{data_in[31][4]}} & 8'h52)^
({{8{data_in[31][5]}} & 8'ha4)^
({{8{data_in[31][6]}} & 8'h63)^
({{8{data_in[31][7]}} & 8'hc6)^
({{8{data_in[32][0]}} & 8'h85)^
({{8{data_in[32][1]}} & 8'h21)^
({{8{data_in[32][2]}} & 8'h42)^
({{8{data_in[32][3]}} & 8'h84)^
({{8{data_in[32][4]}} & 8'h23)^
({{8{data_in[32][5]}} & 8'h46)^
({{8{data_in[32][6]}} & 8'h8c)^
({{8{data_in[32][7]}} & 8'h33)^
({{8{data_in[33][0]}} & 8'h1)^
({{8{data_in[33][1]}} & 8'h2)^
({{8{data_in[33][2]}} & 8'h4)^
({{8{data_in[33][3]}} & 8'h8)^
({{8{data_in[33][4]}} & 8'h10)^
({{8{data_in[33][5]}} & 8'h20)^
({{8{data_in[33][6]}} & 8'h40)^
({{8{data_in[33][7]}} & 8'h80)^
({{8{data_in[34][0]}} & 8'hc0)^
({{8{data_in[34][1]}} & 8'hab)^
({{8{data_in[34][2]}} & 8'h7d)^
({{8{data_in[34][3]}} & 8'hfa)^
({{8{data_in[34][4]}} & 8'hdf)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[34][5]}} & 8'h95)^
({{8{data_in[34][6]}} & 8'h1)^
({{8{data_in[34][7]}} & 8'h2)^
({{8{data_in[35][0]}} & 8'hf4)^
({{8{data_in[35][1]}} & 8'hc3)^
({{8{data_in[35][2]}} & 8'had)^
({{8{data_in[35][3]}} & 8'h71)^
({{8{data_in[35][4]}} & 8'he2)^
({{8{data_in[35][5]}} & 8'hef)^
({{8{data_in[35][6]}} & 8'hf5)^
({{8{data_in[35][7]}} & 8'hc1)^
({{8{data_in[36][0]}} & 8'h4b)^
({{8{data_in[36][1]}} & 8'h96)^
({{8{data_in[36][2]}} & 8'h7)^
({{8{data_in[36][3]}} & 8'he)^
({{8{data_in[36][4]}} & 8'h1c)^
({{8{data_in[36][5]}} & 8'h38)^
({{8{data_in[36][6]}} & 8'h70)^
({{8{data_in[36][7]}} & 8'he0)^
({{8{data_in[37][0]}} & 8'h1f)^
({{8{data_in[37][1]}} & 8'h3e)^
({{8{data_in[37][2]}} & 8'h7c)^
({{8{data_in[37][3]}} & 8'hf8)^
({{8{data_in[37][4]}} & 8'hdb)^
({{8{data_in[37][5]}} & 8'h9d)^
({{8{data_in[37][6]}} & 8'h11)^
({{8{data_in[37][7]}} & 8'h22)^
({{8{data_in[38][0]}} & 8'hda)^
({{8{data_in[38][1]}} & 8'h9f)^
({{8{data_in[38][2]}} & 8'h15)^
({{8{data_in[38][3]}} & 8'h2a)^
({{8{data_in[38][4]}} & 8'h54)^
({{8{data_in[38][5]}} & 8'ha8)^
({{8{data_in[38][6]}} & 8'h7b)^
({{8{data_in[38][7]}} & 8'hf6)^
({{8{data_in[39][0]}} & 8'he8)^
({{8{data_in[39][1]}} & 8'hfb)^
({{8{data_in[39][2]}} & 8'hdd)^
({{8{data_in[39][3]}} & 8'h91)^
({{8{data_in[39][4]}} & 8'h9)^
({{8{data_in[39][5]}} & 8'h12)^
({{8{data_in[39][6]}} & 8'h24)^
({{8{data_in[39][7]}} & 8'h48)^
({{8{data_in[40][0]}} & 8'hfb)^
({{8{data_in[40][1]}} & 8'hdd)^
({{8{data_in[40][2]}} & 8'h91)^
({{8{data_in[40][3]}} & 8'h9)^
({{8{data_in[40][4]}} & 8'h12)^
({{8{data_in[40][5]}} & 8'h24)^
({{8{data_in[40][6]}} & 8'h48)^
({{8{data_in[40][7]}} & 8'h90)^
({{8{data_in[41][0]}} & 8'he4)^
({{8{data_in[41][1]}} & 8'he3)^
({{8{data_in[41][2]}} & 8'hed)^
({{8{data_in[41][3]}} & 8'hf1)^
({{8{data_in[41][4]}} & 8'hc9)^
({{8{data_in[41][5]}} & 8'hb9)^
({{8{data_in[41][6]}} & 8'h59)^
({{8{data_in[41][7]}} & 8'hb2)^
({{8{data_in[42][0]}} & 8'h8a)^
({{8{data_in[42][1]}} & 8'h3f)^
({{8{data_in[42][2]}} & 8'h7e)^
({{8{data_in[42][3]}} & 8'hfc)^
({{8{data_in[42][4]}} & 8'hd3)^
({{8{data_in[42][5]}} & 8'h8d)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[42][6]}} & 8'h31)^
({{8{data_in[42][7]}} & 8'h62)^
({{8{data_in[43][0]}} & 8'h7e)^
({{8{data_in[43][1]}} & 8'hfc)^
({{8{data_in[43][2]}} & 8'hd3)^
({{8{data_in[43][3]}} & 8'h8d)^
({{8{data_in[43][4]}} & 8'h31)^
({{8{data_in[43][5]}} & 8'h62)^
({{8{data_in[43][6]}} & 8'hc4)^
({{8{data_in[43][7]}} & 8'ha3)^
({{8{data_in[44][0]}} & 8'h89)^
({{8{data_in[44][1]}} & 8'h39)^
({{8{data_in[44][2]}} & 8'h72)^
({{8{data_in[44][3]}} & 8'he4)^
({{8{data_in[44][4]}} & 8'he3)^
({{8{data_in[44][5]}} & 8'hed)^
({{8{data_in[44][6]}} & 8'hf1)^
({{8{data_in[44][7]}} & 8'hc9)^
({{8{data_in[45][0]}} & 8'hae)^
({{8{data_in[45][1]}} & 8'h77)^
({{8{data_in[45][2]}} & 8'hee)^
({{8{data_in[45][3]}} & 8'hf7)^
({{8{data_in[45][4]}} & 8'hc5)^
({{8{data_in[45][5]}} & 8'ha1)^
({{8{data_in[45][6]}} & 8'h69)^
({{8{data_in[45][7]}} & 8'hd2)^
({{8{data_in[46][0]}} & 8'h29)^
({{8{data_in[46][1]}} & 8'h52)^
({{8{data_in[46][2]}} & 8'ha4)^
({{8{data_in[46][3]}} & 8'h63)^
({{8{data_in[46][4]}} & 8'hc6)^
({{8{data_in[46][5]}} & 8'ha7)^
({{8{data_in[46][6]}} & 8'h65)^
({{8{data_in[46][7]}} & 8'hca)^
({{8{data_in[47][0]}} & 8'h27)^
({{8{data_in[47][1]}} & 8'h4e)^
({{8{data_in[47][2]}} & 8'h9c)^
({{8{data_in[47][3]}} & 8'h13)^
({{8{data_in[47][4]}} & 8'h26)^
({{8{data_in[47][5]}} & 8'h4c)^
({{8{data_in[47][6]}} & 8'h98)^
({{8{data_in[47][7]}} & 8'h1b)^
({{8{data_in[48][0]}} & 8'ha)^
({{8{data_in[48][1]}} & 8'h14)^
({{8{data_in[48][2]}} & 8'h28)^
({{8{data_in[48][3]}} & 8'h50)^
({{8{data_in[48][4]}} & 8'ha0)^
({{8{data_in[48][5]}} & 8'h6b)^
({{8{data_in[48][6]}} & 8'hd6)^
({{8{data_in[48][7]}} & 8'h87)^
({{8{data_in[49][0]}} & 8'hbb)^
({{8{data_in[49][1]}} & 8'h5d)^
({{8{data_in[49][2]}} & 8'hba)^
({{8{data_in[49][3]}} & 8'h5f)^
({{8{data_in[49][4]}} & 8'hbe)^
({{8{data_in[49][5]}} & 8'h57)^
({{8{data_in[49][6]}} & 8'hae)^
({{8{data_in[49][7]}} & 8'h77)^
({{8{data_in[50][0]}} & 8'hc6)^
({{8{data_in[50][1]}} & 8'ha7)^
({{8{data_in[50][2]}} & 8'h65)^
({{8{data_in[50][3]}} & 8'hca)^
({{8{data_in[50][4]}} & 8'hbf)^
({{8{data_in[50][5]}} & 8'h55)^
({{8{data_in[50][6]}} & 8'haa)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[50][7]}} & 8'h7f)^
({{8{data_in[51][0]}} & 8'he6)^
({{8{data_in[51][1]}} & 8'he7)^
({{8{data_in[51][2]}} & 8'he5)^
({{8{data_in[51][3]}} & 8'he1)^
({{8{data_in[51][4]}} & 8'he9)^
({{8{data_in[51][5]}} & 8'hf9)^
({{8{data_in[51][6]}} & 8'hd9)^
({{8{data_in[51][7]}} & 8'h99)^
({{8{data_in[52][0]}} & 8'h59)^
({{8{data_in[52][1]}} & 8'hb2)^
({{8{data_in[52][2]}} & 8'h4f)^
({{8{data_in[52][3]}} & 8'h9e)^
({{8{data_in[52][4]}} & 8'h17)^
({{8{data_in[52][5]}} & 8'h2e)^
({{8{data_in[52][6]}} & 8'h5c)^
({{8{data_in[52][7]}} & 8'hb8)^
({{8{data_in[53][0]}} & 8'hcd)^
({{8{data_in[53][1]}} & 8'hb1)^
({{8{data_in[53][2]}} & 8'h49)^
({{8{data_in[53][3]}} & 8'h92)^
({{8{data_in[53][4]}} & 8'hf)^
({{8{data_in[53][5]}} & 8'h1e)^
({{8{data_in[53][6]}} & 8'h3c)^
({{8{data_in[53][7]}} & 8'h78)^
({{8{data_in[54][0]}} & 8'ha9)^
({{8{data_in[54][1]}} & 8'h79)^
({{8{data_in[54][2]}} & 8'hf2)^
({{8{data_in[54][3]}} & 8'hcf)^
({{8{data_in[54][4]}} & 8'hb5)^
({{8{data_in[54][5]}} & 8'h41)^
({{8{data_in[54][6]}} & 8'h82)^
({{8{data_in[54][7]}} & 8'h2f)^
({{8{data_in[55][0]}} & 8'hb1)^
({{8{data_in[55][1]}} & 8'h49)^
({{8{data_in[55][2]}} & 8'h92)^
({{8{data_in[55][3]}} & 8'hf)^
({{8{data_in[55][4]}} & 8'h1e)^
({{8{data_in[55][5]}} & 8'h3c)^
({{8{data_in[55][6]}} & 8'h78)^
({{8{data_in[55][7]}} & 8'hf0)^
({{8{data_in[56][0]}} & 8'hbb)^
({{8{data_in[56][1]}} & 8'h5d)^
({{8{data_in[56][2]}} & 8'hba)^
({{8{data_in[56][3]}} & 8'h5f)^
({{8{data_in[56][4]}} & 8'hbe)^
({{8{data_in[56][5]}} & 8'h57)^
({{8{data_in[56][6]}} & 8'hae)^
({{8{data_in[56][7]}} & 8'h77)^
({{8{data_in[57][0]}} & 8'h49)^
({{8{data_in[57][1]}} & 8'h92)^
({{8{data_in[57][2]}} & 8'hf)^
({{8{data_in[57][3]}} & 8'h1e)^
({{8{data_in[57][4]}} & 8'h3c)^
({{8{data_in[57][5]}} & 8'h78)^
({{8{data_in[57][6]}} & 8'hf0)^
({{8{data_in[57][7]}} & 8'hcb)^
({{8{data_in[58][0]}} & 8'hc8)^
({{8{data_in[58][1]}} & 8'hbb)^
({{8{data_in[58][2]}} & 8'h5d)^
({{8{data_in[58][3]}} & 8'hba)^
({{8{data_in[58][4]}} & 8'h5f)^
({{8{data_in[58][5]}} & 8'hbe)^
({{8{data_in[58][6]}} & 8'h57)^
({{8{data_in[58][7]}} & 8'hae)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[59][0]}} & 8'h2f)^
({{8{data_in[59][1]}} & 8'h5e)^
({{8{data_in[59][2]}} & 8'hbc)^
({{8{data_in[59][3]}} & 8'h53)^
({{8{data_in[59][4]}} & 8'ha6)^
({{8{data_in[59][5]}} & 8'h67)^
({{8{data_in[59][6]}} & 8'hce)^
({{8{data_in[59][7]}} & 8'hb7)^
({{8{data_in[60][0]}} & 8'h15)^
({{8{data_in[60][1]}} & 8'h2a)^
({{8{data_in[60][2]}} & 8'h54)^
({{8{data_in[60][3]}} & 8'ha8)^
({{8{data_in[60][4]}} & 8'h7b)^
({{8{data_in[60][5]}} & 8'hf6)^
({{8{data_in[60][6]}} & 8'hc7)^
({{8{data_in[60][7]}} & 8'ha5)^
({{8{data_in[61][0]}} & 8'h42)^
({{8{data_in[61][1]}} & 8'h84)^
({{8{data_in[61][2]}} & 8'h23)^
({{8{data_in[61][3]}} & 8'h46)^
({{8{data_in[61][4]}} & 8'h8c)^
({{8{data_in[61][5]}} & 8'h33)^
({{8{data_in[61][6]}} & 8'h66)^
({{8{data_in[61][7]}} & 8'hcc)^
({{8{data_in[62][0]}} & 8'h6f)^
({{8{data_in[62][1]}} & 8'hde)^
({{8{data_in[62][2]}} & 8'h97)^
({{8{data_in[62][3]}} & 8'h5)^
({{8{data_in[62][4]}} & 8'ha)^
({{8{data_in[62][5]}} & 8'h14)^
({{8{data_in[62][6]}} & 8'h28)^
({{8{data_in[62][7]}} & 8'h50)^
({{8{data_in[63][0]}} & 8'h66)^
({{8{data_in[63][1]}} & 8'hcc)^
({{8{data_in[63][2]}} & 8'hb3)^
({{8{data_in[63][3]}} & 8'h4d)^
({{8{data_in[63][4]}} & 8'h9a)^
({{8{data_in[63][5]}} & 8'h1f)^
({{8{data_in[63][6]}} & 8'h3e)^
({{8{data_in[63][7]}} & 8'h7c)^
({{8{data_in[64][0]}} & 8'h76)^
({{8{data_in[64][1]}} & 8'hec)^
({{8{data_in[64][2]}} & 8'hf3)^
({{8{data_in[64][3]}} & 8'hcd)^
({{8{data_in[64][4]}} & 8'hb1)^
({{8{data_in[64][5]}} & 8'h49)^
({{8{data_in[64][6]}} & 8'h92)^
({{8{data_in[64][7]}} & 8'hf)^
({{8{data_in[65][0]}} & 8'hd)^
({{8{data_in[65][1]}} & 8'h1a)^
({{8{data_in[65][2]}} & 8'h34)^
({{8{data_in[65][3]}} & 8'h68)^
({{8{data_in[65][4]}} & 8'hd0)^
({{8{data_in[65][5]}} & 8'h8b)^
({{8{data_in[65][6]}} & 8'h3d)^
({{8{data_in[65][7]}} & 8'h7a)^
({{8{data_in[66][0]}} & 8'hd0)^
({{8{data_in[66][1]}} & 8'h8b)^
({{8{data_in[66][2]}} & 8'h3d)^
({{8{data_in[66][3]}} & 8'h7a)^
({{8{data_in[66][4]}} & 8'hf4)^
({{8{data_in[66][5]}} & 8'hc3)^
({{8{data_in[66][6]}} & 8'had)^
({{8{data_in[66][7]}} & 8'h71)^
({{8{data_in[67][0]}} & 8'ha2)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[67][1]}} & 8'h6f)^
({{8{data_in[67][2]}} & 8'hde)^
({{8{data_in[67][3]}} & 8'h97)^
({{8{data_in[67][4]}} & 8'h5)^
({{8{data_in[67][5]}} & 8'ha)^
({{8{data_in[67][6]}} & 8'h14)^
({{8{data_in[67][7]}} & 8'h28)^
({{8{data_in[68][0]}} & 8'ha0)^
({{8{data_in[68][1]}} & 8'h6b)^
({{8{data_in[68][2]}} & 8'hd6)^
({{8{data_in[68][3]}} & 8'h87)^
({{8{data_in[68][4]}} & 8'h25)^
({{8{data_in[68][5]}} & 8'h4a)^
({{8{data_in[68][6]}} & 8'h94)^
({{8{data_in[68][7]}} & 8'h3)^
({{8{data_in[69][0]}} & 8'h1e)^
({{8{data_in[69][1]}} & 8'h3c)^
({{8{data_in[69][2]}} & 8'h78)^
({{8{data_in[69][3]}} & 8'hf0)^
({{8{data_in[69][4]}} & 8'hcb)^
({{8{data_in[69][5]}} & 8'hbd)^
({{8{data_in[69][6]}} & 8'h51)^
({{8{data_in[69][7]}} & 8'ha2)^
({{8{data_in[70][0]}} & 8'h7a)^
({{8{data_in[70][1]}} & 8'hf4)^
({{8{data_in[70][2]}} & 8'hc3)^
({{8{data_in[70][3]}} & 8'had)^
({{8{data_in[70][4]}} & 8'h71)^
({{8{data_in[70][5]}} & 8'he2)^
({{8{data_in[70][6]}} & 8'hef)^
({{8{data_in[70][7]}} & 8'hf5)^
({{8{data_in[71][0]}} & 8'h67)^
({{8{data_in[71][1]}} & 8'hce)^
({{8{data_in[71][2]}} & 8'hb7)^
({{8{data_in[71][3]}} & 8'h45)^
({{8{data_in[71][4]}} & 8'h8a)^
({{8{data_in[71][5]}} & 8'h3f)^
({{8{data_in[71][6]}} & 8'h7e)^
({{8{data_in[71][7]}} & 8'hfc)^
({{8{data_in[72][0]}} & 8'h16)^
({{8{data_in[72][1]}} & 8'h2c)^
({{8{data_in[72][2]}} & 8'h58)^
({{8{data_in[72][3]}} & 8'hb0)^
({{8{data_in[72][4]}} & 8'h4b)^
({{8{data_in[72][5]}} & 8'h96)^
({{8{data_in[72][6]}} & 8'h7)^
({{8{data_in[72][7]}} & 8'he)^
({{8{data_in[73][0]}} & 8'hcb)^
({{8{data_in[73][1]}} & 8'hbd)^
({{8{data_in[73][2]}} & 8'h51)^
({{8{data_in[73][3]}} & 8'ha2)^
({{8{data_in[73][4]}} & 8'h6f)^
({{8{data_in[73][5]}} & 8'hde)^
({{8{data_in[73][6]}} & 8'h97)^
({{8{data_in[73][7]}} & 8'h5)^
({{8{data_in[74][0]}} & 8'h1a)^
({{8{data_in[74][1]}} & 8'h34)^
({{8{data_in[74][2]}} & 8'h68)^
({{8{data_in[74][3]}} & 8'hd0)^
({{8{data_in[74][4]}} & 8'h8b)^
({{8{data_in[74][5]}} & 8'h3d)^
({{8{data_in[74][6]}} & 8'h7a)^
({{8{data_in[74][7]}} & 8'hf4)^
({{8{data_in[75][0]}} & 8'h2)^
({{8{data_in[75][1]}} & 8'h4)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[75][2]}} & 8'h8)^
({{8{data_in[75][3]}} & 8'h10)^
({{8{data_in[75][4]}} & 8'h20)^
({{8{data_in[75][5]}} & 8'h40)^
({{8{data_in[75][6]}} & 8'h80)^
({{8{data_in[75][7]}} & 8'h2b)^
({{8{data_in[76][0]}} & 8'h4a)^
({{8{data_in[76][1]}} & 8'h94)^
({{8{data_in[76][2]}} & 8'h3)^
({{8{data_in[76][3]}} & 8'h6)^
({{8{data_in[76][4]}} & 8'hc)^
({{8{data_in[76][5]}} & 8'h18)^
({{8{data_in[76][6]}} & 8'h30)^
({{8{data_in[76][7]}} & 8'h60)^
({{8{data_in[77][0]}} & 8'h7a)^
({{8{data_in[77][1]}} & 8'hf4)^
({{8{data_in[77][2]}} & 8'hc3)^
({{8{data_in[77][3]}} & 8'had)^
({{8{data_in[77][4]}} & 8'h71)^
({{8{data_in[77][5]}} & 8'he2)^
({{8{data_in[77][6]}} & 8'hef)^
({{8{data_in[77][7]}} & 8'hf5)^
({{8{data_in[78][0]}} & 8'hdc)^
({{8{data_in[78][1]}} & 8'h93)^
({{8{data_in[78][2]}} & 8'hd)^
({{8{data_in[78][3]}} & 8'h1a)^
({{8{data_in[78][4]}} & 8'h34)^
({{8{data_in[78][5]}} & 8'h68)^
({{8{data_in[78][6]}} & 8'hd0)^
({{8{data_in[78][7]}} & 8'h8b)^
({{8{data_in[79][0]}} & 8'hd8)^
({{8{data_in[79][1]}} & 8'h9b)^
({{8{data_in[79][2]}} & 8'h1d)^
({{8{data_in[79][3]}} & 8'h3a)^
({{8{data_in[79][4]}} & 8'h74)^
({{8{data_in[79][5]}} & 8'he8)^
({{8{data_in[79][6]}} & 8'hfb)^
({{8{data_in[79][7]}} & 8'hdd)^
({{8{data_in[80][0]}} & 8'h34)^
({{8{data_in[80][1]}} & 8'h68)^
({{8{data_in[80][2]}} & 8'hd0)^
({{8{data_in[80][3]}} & 8'h8b)^
({{8{data_in[80][4]}} & 8'h3d)^
({{8{data_in[80][5]}} & 8'h7a)^
({{8{data_in[80][6]}} & 8'hf4)^
({{8{data_in[80][7]}} & 8'hc3)^
({{8{data_in[81][0]}} & 8'hec)^
({{8{data_in[81][1]}} & 8'hf3)^
({{8{data_in[81][2]}} & 8'hcd)^
({{8{data_in[81][3]}} & 8'hb1)^
({{8{data_in[81][4]}} & 8'h49)^
({{8{data_in[81][5]}} & 8'h92)^
({{8{data_in[81][6]}} & 8'hf)^
({{8{data_in[81][7]}} & 8'h1e)^
({{8{data_in[82][0]}} & 8'hd9)^
({{8{data_in[82][1]}} & 8'h99)^
({{8{data_in[82][2]}} & 8'h19)^
({{8{data_in[82][3]}} & 8'h32)^
({{8{data_in[82][4]}} & 8'h64)^
({{8{data_in[82][5]}} & 8'hc8)^
({{8{data_in[82][6]}} & 8'hbb)^
({{8{data_in[82][7]}} & 8'h5d)^
({{8{data_in[83][0]}} & 8'hee)^
({{8{data_in[83][1]}} & 8'hf7)^
({{8{data_in[83][2]}} & 8'hc5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[83][3]}} & 8'ha1)^
({{8{data_in[83][4]}} & 8'h69)^
({{8{data_in[83][5]}} & 8'hd2)^
({{8{data_in[83][6]}} & 8'h8f)^
({{8{data_in[83][7]}} & 8'h35)^
({{8{data_in[84][0]}} & 8'h89)^
({{8{data_in[84][1]}} & 8'h39)^
({{8{data_in[84][2]}} & 8'h72)^
({{8{data_in[84][3]}} & 8'he4)^
({{8{data_in[84][4]}} & 8'he3)^
({{8{data_in[84][5]}} & 8'hed)^
({{8{data_in[84][6]}} & 8'hf1)^
({{8{data_in[84][7]}} & 8'hc9)^
({{8{data_in[85][0]}} & 8'h56)^
({{8{data_in[85][1]}} & 8'hac)^
({{8{data_in[85][2]}} & 8'h73)^
({{8{data_in[85][3]}} & 8'he6)^
({{8{data_in[85][4]}} & 8'he7)^
({{8{data_in[85][5]}} & 8'he5)^
({{8{data_in[85][6]}} & 8'he1)^
({{8{data_in[85][7]}} & 8'he9)^
({{8{data_in[86][0]}} & 8'h65)^
({{8{data_in[86][1]}} & 8'hca)^
({{8{data_in[86][2]}} & 8'hbf)^
({{8{data_in[86][3]}} & 8'h55)^
({{8{data_in[86][4]}} & 8'haa)^
({{8{data_in[86][5]}} & 8'h7f)^
({{8{data_in[86][6]}} & 8'hfe)^
({{8{data_in[86][7]}} & 8'hd7)^
({{8{data_in[87][0]}} & 8'h68)^
({{8{data_in[87][1]}} & 8'hd0)^
({{8{data_in[87][2]}} & 8'h8b)^
({{8{data_in[87][3]}} & 8'h3d)^
({{8{data_in[87][4]}} & 8'h7a)^
({{8{data_in[87][5]}} & 8'hf4)^
({{8{data_in[87][6]}} & 8'hc3)^
({{8{data_in[87][7]}} & 8'had)^
({{8{data_in[88][0]}} & 8'hec)^
({{8{data_in[88][1]}} & 8'hf3)^
({{8{data_in[88][2]}} & 8'hcd)^
({{8{data_in[88][3]}} & 8'hb1)^
({{8{data_in[88][4]}} & 8'h49)^
({{8{data_in[88][5]}} & 8'h92)^
({{8{data_in[88][6]}} & 8'hf)^
({{8{data_in[88][7]}} & 8'h1e)^
({{8{data_in[89][0]}} & 8'he8)^
({{8{data_in[89][1]}} & 8'hfb)^
({{8{data_in[89][2]}} & 8'hdd)^
({{8{data_in[89][3]}} & 8'h91)^
({{8{data_in[89][4]}} & 8'h9)^
({{8{data_in[89][5]}} & 8'h12)^
({{8{data_in[89][6]}} & 8'h24)^
({{8{data_in[89][7]}} & 8'h48)^
({{8{data_in[90][0]}} & 8'h5)^
({{8{data_in[90][1]}} & 8'ha)^
({{8{data_in[90][2]}} & 8'h14)^
({{8{data_in[90][3]}} & 8'h28)^
({{8{data_in[90][4]}} & 8'h50)^
({{8{data_in[90][5]}} & 8'ha0)^
({{8{data_in[90][6]}} & 8'h6b)^
({{8{data_in[90][7]}} & 8'hd6)^
({{8{data_in[91][0]}} & 8'h6)^
({{8{data_in[91][1]}} & 8'hc)^
({{8{data_in[91][2]}} & 8'h18)^
({{8{data_in[91][3]}} & 8'h30)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[91][4]}} & 8'h60)^
({{8{data_in[91][5]}} & 8'hc0)^
({{8{data_in[91][6]}} & 8'hab)^
({{8{data_in[91][7]}} & 8'h7d)^
({{8{data_in[92][0]}} & 8'h38)^
({{8{data_in[92][1]}} & 8'h70)^
({{8{data_in[92][2]}} & 8'he0)^
({{8{data_in[92][3]}} & 8'heb)^
({{8{data_in[92][4]}} & 8'hfd)^
({{8{data_in[92][5]}} & 8'hd1)^
({{8{data_in[92][6]}} & 8'h89)^
({{8{data_in[92][7]}} & 8'h39)^
({{8{data_in[93][0]}} & 8'haf)^
({{8{data_in[93][1]}} & 8'h75)^
({{8{data_in[93][2]}} & 8'hea)^
({{8{data_in[93][3]}} & 8'hff)^
({{8{data_in[93][4]}} & 8'hd5)^
({{8{data_in[93][5]}} & 8'h81)^
({{8{data_in[93][6]}} & 8'h29)^
({{8{data_in[93][7]}} & 8'h52)^
({{8{data_in[94][0]}} & 8'h66)^
({{8{data_in[94][1]}} & 8'hcc)^
({{8{data_in[94][2]}} & 8'hb3)^
({{8{data_in[94][3]}} & 8'h4d)^
({{8{data_in[94][4]}} & 8'h9a)^
({{8{data_in[94][5]}} & 8'h1f)^
({{8{data_in[94][6]}} & 8'h3e)^
({{8{data_in[94][7]}} & 8'h7c)^
({{8{data_in[95][0]}} & 8'hc4)^
({{8{data_in[95][1]}} & 8'ha3)^
({{8{data_in[95][2]}} & 8'h6d)^
({{8{data_in[95][3]}} & 8'hda)^
({{8{data_in[95][4]}} & 8'h9f)^
({{8{data_in[95][5]}} & 8'h15)^
({{8{data_in[95][6]}} & 8'h2a)^
({{8{data_in[95][7]}} & 8'h54)^
({{8{data_in[96][0]}} & 8'hdd)^
({{8{data_in[96][1]}} & 8'h91)^
({{8{data_in[96][2]}} & 8'h9)^
({{8{data_in[96][3]}} & 8'h12)^
({{8{data_in[96][4]}} & 8'h24)^
({{8{data_in[96][5]}} & 8'h48)^
({{8{data_in[96][6]}} & 8'h90)^
({{8{data_in[96][7]}} & 8'hb)^
({{8{data_in[97][0]}} & 8'h20)^
({{8{data_in[97][1]}} & 8'h40)^
({{8{data_in[97][2]}} & 8'h80)^
({{8{data_in[97][3]}} & 8'h2b)^
({{8{data_in[97][4]}} & 8'h56)^
({{8{data_in[97][5]}} & 8'hac)^
({{8{data_in[97][6]}} & 8'h73)^
({{8{data_in[97][7]}} & 8'he6)^
({{8{data_in[98][0]}} & 8'h3c)^
({{8{data_in[98][1]}} & 8'h78)^
({{8{data_in[98][2]}} & 8'hf0)^
({{8{data_in[98][3]}} & 8'hcb)^
({{8{data_in[98][4]}} & 8'hbd)^
({{8{data_in[98][5]}} & 8'h51)^
({{8{data_in[98][6]}} & 8'ha2)^
({{8{data_in[98][7]}} & 8'h6f)^
({{8{data_in[99][0]}} & 8'hcf)^
({{8{data_in[99][1]}} & 8'hb5)^
({{8{data_in[99][2]}} & 8'h41)^
({{8{data_in[99][3]}} & 8'h82)^
({{8{data_in[99][4]}} & 8'h2f)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[99][5]}} & 8'h5e)^
({{8{data_in[99][6]}} & 8'hbc)^
({{8{data_in[99][7]}} & 8'h53)^
({{8{data_in[100][0]}} & 8'h51)^
({{8{data_in[100][1]}} & 8'ha2)^
({{8{data_in[100][2]}} & 8'h6f)^
({{8{data_in[100][3]}} & 8'hde)^
({{8{data_in[100][4]}} & 8'h97)^
({{8{data_in[100][5]}} & 8'h5)^
({{8{data_in[100][6]}} & 8'ha)^
({{8{data_in[100][7]}} & 8'h14)^
({{8{data_in[101][0]}} & 8'he)^
({{8{data_in[101][1]}} & 8'h1c)^
({{8{data_in[101][2]}} & 8'h38)^
({{8{data_in[101][3]}} & 8'h70)^
({{8{data_in[101][4]}} & 8'he0)^
({{8{data_in[101][5]}} & 8'heb)^
({{8{data_in[101][6]}} & 8'hfd)^
({{8{data_in[101][7]}} & 8'hd1)^
({{8{data_in[102][0]}} & 8'h16)^
({{8{data_in[102][1]}} & 8'h2c)^
({{8{data_in[102][2]}} & 8'h58)^
({{8{data_in[102][3]}} & 8'hb0)^
({{8{data_in[102][4]}} & 8'h4b)^
({{8{data_in[102][5]}} & 8'h96)^
({{8{data_in[102][6]}} & 8'h7)^
({{8{data_in[102][7]}} & 8'he)^
({{8{data_in[103][0]}} & 8'hb4)^
({{8{data_in[103][1]}} & 8'h43)^
({{8{data_in[103][2]}} & 8'h86)^
({{8{data_in[103][3]}} & 8'h27)^
({{8{data_in[103][4]}} & 8'h4e)^
({{8{data_in[103][5]}} & 8'h9c)^
({{8{data_in[103][6]}} & 8'h13)^
({{8{data_in[103][7]}} & 8'h26)^
({{8{data_in[104][0]}} & 8'h31)^
({{8{data_in[104][1]}} & 8'h62)^
({{8{data_in[104][2]}} & 8'hc4)^
({{8{data_in[104][3]}} & 8'ha3)^
({{8{data_in[104][4]}} & 8'h6d)^
({{8{data_in[104][5]}} & 8'hda)^
({{8{data_in[104][6]}} & 8'h9f)^
({{8{data_in[104][7]}} & 8'h15)^
({{8{data_in[105][0]}} & 8'hf0)^
({{8{data_in[105][1]}} & 8'hcb)^
({{8{data_in[105][2]}} & 8'hbd)^
({{8{data_in[105][3]}} & 8'h51)^
({{8{data_in[105][4]}} & 8'ha2)^
({{8{data_in[105][5]}} & 8'h6f)^
({{8{data_in[105][6]}} & 8'hde)^
({{8{data_in[105][7]}} & 8'h97)^
({{8{data_in[106][0]}} & 8'h92)^
({{8{data_in[106][1]}} & 8'hf)^
({{8{data_in[106][2]}} & 8'h1e)^
({{8{data_in[106][3]}} & 8'h3c)^
({{8{data_in[106][4]}} & 8'h78)^
({{8{data_in[106][5]}} & 8'hf0)^
({{8{data_in[106][6]}} & 8'hcb)^
({{8{data_in[106][7]}} & 8'hbd)^
({{8{data_in[107][0]}} & 8'hf5)^
({{8{data_in[107][1]}} & 8'hc1)^
({{8{data_in[107][2]}} & 8'ha9)^
({{8{data_in[107][3]}} & 8'h79)^
({{8{data_in[107][4]}} & 8'hf2)^
({{8{data_in[107][5]}} & 8'hcf)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[107][6]}} & 8'hb5)^
({{8{data_in[107][7]}} & 8'h41)^
({{8{data_in[108][0]}} & 8'h84)^
({{8{data_in[108][1]}} & 8'h23)^
({{8{data_in[108][2]}} & 8'h46)^
({{8{data_in[108][3]}} & 8'h8c)^
({{8{data_in[108][4]}} & 8'h33)^
({{8{data_in[108][5]}} & 8'h66)^
({{8{data_in[108][6]}} & 8'hcc)^
({{8{data_in[108][7]}} & 8'hb3)^
({{8{data_in[109][0]}} & 8'h36)^
({{8{data_in[109][1]}} & 8'h6c)^
({{8{data_in[109][2]}} & 8'hd8)^
({{8{data_in[109][3]}} & 8'h9b)^
({{8{data_in[109][4]}} & 8'h1d)^
({{8{data_in[109][5]}} & 8'h3a)^
({{8{data_in[109][6]}} & 8'h74)^
({{8{data_in[109][7]}} & 8'he8)^
({{8{data_in[110][0]}} & 8'h64)^
({{8{data_in[110][1]}} & 8'hc8)^
({{8{data_in[110][2]}} & 8'hbb)^
({{8{data_in[110][3]}} & 8'h5d)^
({{8{data_in[110][4]}} & 8'hba)^
({{8{data_in[110][5]}} & 8'h5f)^
({{8{data_in[110][6]}} & 8'hbe)^
({{8{data_in[110][7]}} & 8'h57)^
({{8{data_in[111][0]}} & 8'hac)^
({{8{data_in[111][1]}} & 8'h73)^
({{8{data_in[111][2]}} & 8'he6)^
({{8{data_in[111][3]}} & 8'he7)^
({{8{data_in[111][4]}} & 8'he5)^
({{8{data_in[111][5]}} & 8'he1)^
({{8{data_in[111][6]}} & 8'he9)^
({{8{data_in[111][7]}} & 8'hf9)^
({{8{data_in[112][0]}} & 8'hf8)^
({{8{data_in[112][1]}} & 8'hdb)^
({{8{data_in[112][2]}} & 8'h9d)^
({{8{data_in[112][3]}} & 8'h11)^
({{8{data_in[112][4]}} & 8'h22)^
({{8{data_in[112][5]}} & 8'h44)^
({{8{data_in[112][6]}} & 8'h88)^
({{8{data_in[112][7]}} & 8'h3b)^
({{8{data_in[113][0]}} & 8'h80)^
({{8{data_in[113][1]}} & 8'h2b)^
({{8{data_in[113][2]}} & 8'h56)^
({{8{data_in[113][3]}} & 8'hac)^
({{8{data_in[113][4]}} & 8'h73)^
({{8{data_in[113][5]}} & 8'he6)^
({{8{data_in[113][6]}} & 8'he7)^
({{8{data_in[113][7]}} & 8'he5)^
({{8{data_in[114][0]}} & 8'hec)^
({{8{data_in[114][1]}} & 8'hf3)^
({{8{data_in[114][2]}} & 8'hcd)^
({{8{data_in[114][3]}} & 8'hb1)^
({{8{data_in[114][4]}} & 8'h49)^
({{8{data_in[114][5]}} & 8'h92)^
({{8{data_in[114][6]}} & 8'hf)^
({{8{data_in[114][7]}} & 8'h1e)^
({{8{data_in[115][0]}} & 8'hb)^
({{8{data_in[115][1]}} & 8'h16)^
({{8{data_in[115][2]}} & 8'h2c)^
({{8{data_in[115][3]}} & 8'h58)^
({{8{data_in[115][4]}} & 8'hb0)^
({{8{data_in[115][5]}} & 8'h4b)^
({{8{data_in[115][6]}} & 8'h96)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[115][7]}} & 8'h7)^
({{8{data_in[116][0]}} & 8'hd8)^
({{8{data_in[116][1]}} & 8'h9b)^
({{8{data_in[116][2]}} & 8'h1d)^
({{8{data_in[116][3]}} & 8'h3a)^
({{8{data_in[116][4]}} & 8'h74)^
({{8{data_in[116][5]}} & 8'he8)^
({{8{data_in[116][6]}} & 8'hfb)^
({{8{data_in[116][7]}} & 8'hdd)^
({{8{data_in[117][0]}} & 8'h80)^
({{8{data_in[117][1]}} & 8'h2b)^
({{8{data_in[117][2]}} & 8'h56)^
({{8{data_in[117][3]}} & 8'hac)^
({{8{data_in[117][4]}} & 8'h73)^
({{8{data_in[117][5]}} & 8'he6)^
({{8{data_in[117][6]}} & 8'he7)^
({{8{data_in[117][7]}} & 8'he5)^
({{8{data_in[118][0]}} & 8'h79)^
({{8{data_in[118][1]}} & 8'hf2)^
({{8{data_in[118][2]}} & 8'hcf)^
({{8{data_in[118][3]}} & 8'hb5)^
({{8{data_in[118][4]}} & 8'h41)^
({{8{data_in[118][5]}} & 8'h82)^
({{8{data_in[118][6]}} & 8'h2f)^
({{8{data_in[118][7]}} & 8'h5e)^
({{8{data_in[119][0]}} & 8'hd5)^
({{8{data_in[119][1]}} & 8'h81)^
({{8{data_in[119][2]}} & 8'h29)^
({{8{data_in[119][3]}} & 8'h52)^
({{8{data_in[119][4]}} & 8'ha4)^
({{8{data_in[119][5]}} & 8'h63)^
({{8{data_in[119][6]}} & 8'hc6)^
({{8{data_in[119][7]}} & 8'ha7)^
({{8{data_in[120][0]}} & 8'h63)^
({{8{data_in[120][1]}} & 8'hc6)^
({{8{data_in[120][2]}} & 8'ha7)^
({{8{data_in[120][3]}} & 8'h65)^
({{8{data_in[120][4]}} & 8'hca)^
({{8{data_in[120][5]}} & 8'hbf)^
({{8{data_in[120][6]}} & 8'h55)^
({{8{data_in[120][7]}} & 8'haa)^
({{8{data_in[121][0]}} & 8'h98)^
({{8{data_in[121][1]}} & 8'h1b)^
({{8{data_in[121][2]}} & 8'h36)^
({{8{data_in[121][3]}} & 8'h6c)^
({{8{data_in[121][4]}} & 8'hd8)^
({{8{data_in[121][5]}} & 8'h9b)^
({{8{data_in[121][6]}} & 8'h1d)^
({{8{data_in[121][7]}} & 8'h3a)^
({{8{data_in[122][0]}} & 8'hf1)^
({{8{data_in[122][1]}} & 8'hc9)^
({{8{data_in[122][2]}} & 8'hb9)^
({{8{data_in[122][3]}} & 8'h59)^
({{8{data_in[122][4]}} & 8'hb2)^
({{8{data_in[122][5]}} & 8'h4f)^
({{8{data_in[122][6]}} & 8'h9e)^
({{8{data_in[122][7]}} & 8'h17)^
({{8{data_in[123][0]}} & 8'h5f)^
({{8{data_in[123][1]}} & 8'hbe)^
({{8{data_in[123][2]}} & 8'h57)^
({{8{data_in[123][3]}} & 8'hae)^
({{8{data_in[123][4]}} & 8'h77)^
({{8{data_in[123][5]}} & 8'hee)^
({{8{data_in[123][6]}} & 8'hf7)^
({{8{data_in[123][7]}} & 8'hc5)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[124][0]}} & 8'h7)^
({{8{data_in[124][1]}} & 8'he)^
({{8{data_in[124][2]}} & 8'h1c)^
({{8{data_in[124][3]}} & 8'h38)^
({{8{data_in[124][4]}} & 8'h70)^
({{8{data_in[124][5]}} & 8'he0)^
({{8{data_in[124][6]}} & 8'heb)^
({{8{data_in[124][7]}} & 8'hfd)^
({{8{data_in[125][0]}} & 8'h38)^
({{8{data_in[125][1]}} & 8'h70)^
({{8{data_in[125][2]}} & 8'he0)^
({{8{data_in[125][3]}} & 8'heb)^
({{8{data_in[125][4]}} & 8'hfd)^
({{8{data_in[125][5]}} & 8'hd1)^
({{8{data_in[125][6]}} & 8'h89)^
({{8{data_in[125][7]}} & 8'h39)^
({{8{data_in[126][0]}} & 8'h47)^
({{8{data_in[126][1]}} & 8'h8e)^
({{8{data_in[126][2]}} & 8'h37)^
({{8{data_in[126][3]}} & 8'h6e)^
({{8{data_in[126][4]}} & 8'hdc)^
({{8{data_in[126][5]}} & 8'h93)^
({{8{data_in[126][6]}} & 8'hd)^
({{8{data_in[126][7]}} & 8'h1a)^
({{8{data_in[127][0]}} & 8'h14)^
({{8{data_in[127][1]}} & 8'h28)^
({{8{data_in[127][2]}} & 8'h50)^
({{8{data_in[127][3]}} & 8'ha0)^
({{8{data_in[127][4]}} & 8'h6b)^
({{8{data_in[127][5]}} & 8'hd6)^
({{8{data_in[127][6]}} & 8'h87)^
({{8{data_in[127][7]}} & 8'h25)^
({{8{data_in[128][0]}} & 8'h54)^
({{8{data_in[128][1]}} & 8'ha8)^
({{8{data_in[128][2]}} & 8'h7b)^
({{8{data_in[128][3]}} & 8'hf6)^
({{8{data_in[128][4]}} & 8'hc7)^
({{8{data_in[128][5]}} & 8'ha5)^
({{8{data_in[128][6]}} & 8'h61)^
({{8{data_in[128][7]}} & 8'hc2)^
({{8{data_in[129][0]}} & 8'h94)^
({{8{data_in[129][1]}} & 8'h3)^
({{8{data_in[129][2]}} & 8'h6)^
({{8{data_in[129][3]}} & 8'hc)^
({{8{data_in[129][4]}} & 8'h18)^
({{8{data_in[129][5]}} & 8'h30)^
({{8{data_in[129][6]}} & 8'h60)^
({{8{data_in[129][7]}} & 8'hc0)^
({{8{data_in[130][0]}} & 8'hd6)^
({{8{data_in[130][1]}} & 8'h87)^
({{8{data_in[130][2]}} & 8'h25)^
({{8{data_in[130][3]}} & 8'h4a)^
({{8{data_in[130][4]}} & 8'h94)^
({{8{data_in[130][5]}} & 8'h3)^
({{8{data_in[130][6]}} & 8'h6)^
({{8{data_in[130][7]}} & 8'hc)^
({{8{data_in[131][0]}} & 8'h9f)^
({{8{data_in[131][1]}} & 8'h15)^
({{8{data_in[131][2]}} & 8'h2a)^
({{8{data_in[131][3]}} & 8'h54)^
({{8{data_in[131][4]}} & 8'ha8)^
({{8{data_in[131][5]}} & 8'h7b)^
({{8{data_in[131][6]}} & 8'hf6)^
({{8{data_in[131][7]}} & 8'hc7)^
({{8{data_in[132][0]}} & 8'h83)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[132][1]}} & 8'h2d)^
({{8{data_in[132][2]}} & 8'h5a)^
({{8{data_in[132][3]}} & 8'hb4)^
({{8{data_in[132][4]}} & 8'h43)^
({{8{data_in[132][5]}} & 8'h86)^
({{8{data_in[132][6]}} & 8'h27)^
({{8{data_in[132][7]}} & 8'h4e)^
({{8{data_in[133][0]}} & 8'h3a)^
({{8{data_in[133][1]}} & 8'h74)^
({{8{data_in[133][2]}} & 8'he8)^
({{8{data_in[133][3]}} & 8'hfb)^
({{8{data_in[133][4]}} & 8'hdd)^
({{8{data_in[133][5]}} & 8'h91)^
({{8{data_in[133][6]}} & 8'h9)^
({{8{data_in[133][7]}} & 8'h12)^
({{8{data_in[134][0]}} & 8'h9b)^
({{8{data_in[134][1]}} & 8'h1d)^
({{8{data_in[134][2]}} & 8'h3a)^
({{8{data_in[134][3]}} & 8'h74)^
({{8{data_in[134][4]}} & 8'he8)^
({{8{data_in[134][5]}} & 8'hfb)^
({{8{data_in[134][6]}} & 8'hdd)^
({{8{data_in[134][7]}} & 8'h91)^
({{8{data_in[135][0]}} & 8'h66)^
({{8{data_in[135][1]}} & 8'hcc)^
({{8{data_in[135][2]}} & 8'hb3)^
({{8{data_in[135][3]}} & 8'h4d)^
({{8{data_in[135][4]}} & 8'h9a)^
({{8{data_in[135][5]}} & 8'h1f)^
({{8{data_in[135][6]}} & 8'h3e)^
({{8{data_in[135][7]}} & 8'h7c)^
({{8{data_in[136][0]}} & 8'hd)^
({{8{data_in[136][1]}} & 8'hf1)^
({{8{data_in[136][2]}} & 8'hc9)^
({{8{data_in[136][3]}} & 8'hb9)^
({{8{data_in[136][4]}} & 8'h59)^
({{8{data_in[136][5]}} & 8'hb2)^
({{8{data_in[136][6]}} & 8'h4f)^
({{8{data_in[136][7]}} & 8'h9e)^
({{8{data_in[137][0]}} & 8'h81)^
({{8{data_in[137][1]}} & 8'h29)^
({{8{data_in[137][2]}} & 8'h52)^
({{8{data_in[137][3]}} & 8'ha4)^
({{8{data_in[137][4]}} & 8'h63)^
({{8{data_in[137][5]}} & 8'hc6)^
({{8{data_in[137][6]}} & 8'ha7)^
({{8{data_in[137][7]}} & 8'h65)^
({{8{data_in[138][0]}} & 8'h7c)^
({{8{data_in[138][1]}} & 8'hf8)^
({{8{data_in[138][2]}} & 8'hdb)^
({{8{data_in[138][3]}} & 8'h9d)^
({{8{data_in[138][4]}} & 8'h11)^
({{8{data_in[138][5]}} & 8'h22)^
({{8{data_in[138][6]}} & 8'h44)^
({{8{data_in[138][7]}} & 8'h88)^
({{8{data_in[139][0]}} & 8'h95)^
({{8{data_in[139][1]}} & 8'h1)^
({{8{data_in[139][2]}} & 8'h2)^
({{8{data_in[139][3]}} & 8'h4)^
({{8{data_in[139][4]}} & 8'h8)^
({{8{data_in[139][5]}} & 8'h10)^
({{8{data_in[139][6]}} & 8'h20)^
({{8{data_in[139][7]}} & 8'h40)^
({{8{data_in[140][0]}} & 8'h96)^
({{8{data_in[140][1]}} & 8'h7)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[140][2]}} & 8'he)^
({{8{data_in[140][3]}} & 8'h1c)^
({{8{data_in[140][4]}} & 8'h38)^
({{8{data_in[140][5]}} & 8'h70)^
({{8{data_in[140][6]}} & 8'he0)^
({{8{data_in[140][7]}} & 8'heb)^
({{8{data_in[141][0]}} & 8'h93)^
({{8{data_in[141][1]}} & 8'hd)^
({{8{data_in[141][2]}} & 8'h1a)^
({{8{data_in[141][3]}} & 8'h34)^
({{8{data_in[141][4]}} & 8'h68)^
({{8{data_in[141][5]}} & 8'hd0)^
({{8{data_in[141][6]}} & 8'h8b)^
({{8{data_in[141][7]}} & 8'h3d)^
({{8{data_in[142][0]}} & 8'h58)^
({{8{data_in[142][1]}} & 8'hb0)^
({{8{data_in[142][2]}} & 8'h4b)^
({{8{data_in[142][3]}} & 8'h96)^
({{8{data_in[142][4]}} & 8'h7)^
({{8{data_in[142][5]}} & 8'he)^
({{8{data_in[142][6]}} & 8'h1c)^
({{8{data_in[142][7]}} & 8'h38)^
({{8{data_in[143][0]}} & 8'h50)^
({{8{data_in[143][1]}} & 8'ha0)^
({{8{data_in[143][2]}} & 8'h6b)^
({{8{data_in[143][3]}} & 8'hd6)^
({{8{data_in[143][4]}} & 8'h87)^
({{8{data_in[143][5]}} & 8'h25)^
({{8{data_in[143][6]}} & 8'h4a)^
({{8{data_in[143][7]}} & 8'h94)^
({{8{data_in[144][0]}} & 8'hb5)^
({{8{data_in[144][1]}} & 8'h41)^
({{8{data_in[144][2]}} & 8'h82)^
({{8{data_in[144][3]}} & 8'h2f)^
({{8{data_in[144][4]}} & 8'h5e)^
({{8{data_in[144][5]}} & 8'hbc)^
({{8{data_in[144][6]}} & 8'h53)^
({{8{data_in[144][7]}} & 8'ha6)^
({{8{data_in[145][0]}} & 8'hdc)^
({{8{data_in[145][1]}} & 8'h93)^
({{8{data_in[145][2]}} & 8'hd)^
({{8{data_in[145][3]}} & 8'h1a)^
({{8{data_in[145][4]}} & 8'h34)^
({{8{data_in[145][5]}} & 8'h68)^
({{8{data_in[145][6]}} & 8'hd0)^
({{8{data_in[145][7]}} & 8'h8b)^
({{8{data_in[146][0]}} & 8'h21)^
({{8{data_in[146][1]}} & 8'h42)^
({{8{data_in[146][2]}} & 8'h84)^
({{8{data_in[146][3]}} & 8'h23)^
({{8{data_in[146][4]}} & 8'h46)^
({{8{data_in[146][5]}} & 8'h8c)^
({{8{data_in[146][6]}} & 8'h33)^
({{8{data_in[146][7]}} & 8'h66)^
({{8{data_in[147][0]}} & 8'h58)^
({{8{data_in[147][1]}} & 8'hb0)^
({{8{data_in[147][2]}} & 8'h4b)^
({{8{data_in[147][3]}} & 8'h96)^
({{8{data_in[147][4]}} & 8'h7)^
({{8{data_in[147][5]}} & 8'he)^
({{8{data_in[147][6]}} & 8'h1c)^
({{8{data_in[147][7]}} & 8'h38)^
({{8{data_in[148][0]}} & 8'hcb)^
({{8{data_in[148][1]}} & 8'hbd)^
({{8{data_in[148][2]}} & 8'h51)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[148][3]}} & 8'ha2)^
({{8{data_in[148][4]}} & 8'h6f)^
({{8{data_in[148][5]}} & 8'hde)^
({{8{data_in[148][6]}} & 8'h97)^
({{8{data_in[148][7]}} & 8'h5)^
({{8{data_in[149][0]}} & 8'hbb)^
({{8{data_in[149][1]}} & 8'h5d)^
({{8{data_in[149][2]}} & 8'hba)^
({{8{data_in[149][3]}} & 8'h5f)^
({{8{data_in[149][4]}} & 8'hbe)^
({{8{data_in[149][5]}} & 8'h57)^
({{8{data_in[149][6]}} & 8'hae)^
({{8{data_in[149][7]}} & 8'h77)^
({{8{data_in[150][0]}} & 8'h4f)^
({{8{data_in[150][1]}} & 8'h9e)^
({{8{data_in[150][2]}} & 8'h17)^
({{8{data_in[150][3]}} & 8'h2e)^
({{8{data_in[150][4]}} & 8'h5c)^
({{8{data_in[150][5]}} & 8'hb8)^
({{8{data_in[150][6]}} & 8'h5b)^
({{8{data_in[150][7]}} & 8'hb6)^
({{8{data_in[151][0]}} & 8'h75)^
({{8{data_in[151][1]}} & 8'hea)^
({{8{data_in[151][2]}} & 8'hff)^
({{8{data_in[151][3]}} & 8'hd5)^
({{8{data_in[151][4]}} & 8'h81)^
({{8{data_in[151][5]}} & 8'h29)^
({{8{data_in[151][6]}} & 8'h52)^
({{8{data_in[151][7]}} & 8'ha4)^
({{8{data_in[152][0]}} & 8'hb8)^
({{8{data_in[152][1]}} & 8'h5b)^
({{8{data_in[152][2]}} & 8'hb6)^
({{8{data_in[152][3]}} & 8'h47)^
({{8{data_in[152][4]}} & 8'h8e)^
({{8{data_in[152][5]}} & 8'h37)^
({{8{data_in[152][6]}} & 8'h6e)^
({{8{data_in[152][7]}} & 8'hdc)^
({{8{data_in[153][0]}} & 8'h1c)^
({{8{data_in[153][1]}} & 8'h38)^
({{8{data_in[153][2]}} & 8'h70)^
({{8{data_in[153][3]}} & 8'he0)^
({{8{data_in[153][4]}} & 8'heb)^
({{8{data_in[153][5]}} & 8'hfd)^
({{8{data_in[153][6]}} & 8'hd1)^
({{8{data_in[153][7]}} & 8'h89)^
({{8{data_in[154][0]}} & 8'he6)^
({{8{data_in[154][1]}} & 8'he7)^
({{8{data_in[154][2]}} & 8'he5)^
({{8{data_in[154][3]}} & 8'he1)^
({{8{data_in[154][4]}} & 8'he9)^
({{8{data_in[154][5]}} & 8'hf9)^
({{8{data_in[154][6]}} & 8'hd9)^
({{8{data_in[154][7]}} & 8'h99)^
({{8{data_in[155][0]}} & 8'h54)^
({{8{data_in[155][1]}} & 8'ha8)^
({{8{data_in[155][2]}} & 8'h7b)^
({{8{data_in[155][3]}} & 8'hf6)^
({{8{data_in[155][4]}} & 8'hc7)^
({{8{data_in[155][5]}} & 8'ha5)^
({{8{data_in[155][6]}} & 8'h61)^
({{8{data_in[155][7]}} & 8'hc2)^
({{8{data_in[156][0]}} & 8'hf1)^
({{8{data_in[156][1]}} & 8'hc9)^
({{8{data_in[156][2]}} & 8'hb9)^
({{8{data_in[156][3]}} & 8'h59)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[156][4]}} & 8'hb2)^
({{8{data_in[156][5]}} & 8'h4f)^
({{8{data_in[156][6]}} & 8'h9e)^
({{8{data_in[156][7]}} & 8'h17)^
({{8{data_in[157][0]}} & 8'hbc)^
({{8{data_in[157][1]}} & 8'h53)^
({{8{data_in[157][2]}} & 8'ha6)^
({{8{data_in[157][3]}} & 8'h67)^
({{8{data_in[157][4]}} & 8'hce)^
({{8{data_in[157][5]}} & 8'hb7)^
({{8{data_in[157][6]}} & 8'h45)^
({{8{data_in[157][7]}} & 8'h8a)^
({{8{data_in[158][0]}} & 8'hbc)^
({{8{data_in[158][1]}} & 8'h53)^
({{8{data_in[158][2]}} & 8'ha6)^
({{8{data_in[158][3]}} & 8'h67)^
({{8{data_in[158][4]}} & 8'hce)^
({{8{data_in[158][5]}} & 8'hb7)^
({{8{data_in[158][6]}} & 8'h45)^
({{8{data_in[158][7]}} & 8'h8a)^
({{8{data_in[159][0]}} & 8'hab)^
({{8{data_in[159][1]}} & 8'h7d)^
({{8{data_in[159][2]}} & 8'hfa)^
({{8{data_in[159][3]}} & 8'hdf)^
({{8{data_in[159][4]}} & 8'h95)^
({{8{data_in[159][5]}} & 8'h1)^
({{8{data_in[159][6]}} & 8'h2)^
({{8{data_in[159][7]}} & 8'h4)^
({{8{data_in[160][0]}} & 8'h2)^
({{8{data_in[160][1]}} & 8'h4)^
({{8{data_in[160][2]}} & 8'h8)^
({{8{data_in[160][3]}} & 8'h10)^
({{8{data_in[160][4]}} & 8'h20)^
({{8{data_in[160][5]}} & 8'h40)^
({{8{data_in[160][6]}} & 8'h80)^
({{8{data_in[160][7]}} & 8'h2b)^
({{8{data_in[161][0]}} & 8'h5b)^
({{8{data_in[161][1]}} & 8'hb6)^
({{8{data_in[161][2]}} & 8'h47)^
({{8{data_in[161][3]}} & 8'h8e)^
({{8{data_in[161][4]}} & 8'h37)^
({{8{data_in[161][5]}} & 8'h6e)^
({{8{data_in[161][6]}} & 8'hdc)^
({{8{data_in[161][7]}} & 8'h93)^
({{8{data_in[162][0]}} & 8'h9d)^
({{8{data_in[162][1]}} & 8'h11)^
({{8{data_in[162][2]}} & 8'h22)^
({{8{data_in[162][3]}} & 8'h44)^
({{8{data_in[162][4]}} & 8'h88)^
({{8{data_in[162][5]}} & 8'h3b)^
({{8{data_in[162][6]}} & 8'h76)^
({{8{data_in[162][7]}} & 8'hec)^
({{8{data_in[163][0]}} & 8'h43)^
({{8{data_in[163][1]}} & 8'h86)^
({{8{data_in[163][2]}} & 8'h27)^
({{8{data_in[163][3]}} & 8'h4e)^
({{8{data_in[163][4]}} & 8'h9c)^
({{8{data_in[163][5]}} & 8'h13)^
({{8{data_in[163][6]}} & 8'h26)^
({{8{data_in[163][7]}} & 8'h4c)^
({{8{data_in[164][0]}} & 8'h5b)^
({{8{data_in[164][1]}} & 8'hb6)^
({{8{data_in[164][2]}} & 8'h47)^
({{8{data_in[164][3]}} & 8'h8e)^
({{8{data_in[164][4]}} & 8'h37)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[164][5]}} & 8'h6e)^
({{8{data_in[164][6]}} & 8'hdc)^
({{8{data_in[164][7]}} & 8'h93)^
({{8{data_in[165][0]}} & 8'h58)^
({{8{data_in[165][1]}} & 8'hb0)^
({{8{data_in[165][2]}} & 8'h4b)^
({{8{data_in[165][3]}} & 8'h96)^
({{8{data_in[165][4]}} & 8'h7)^
({{8{data_in[165][5]}} & 8'he)^
({{8{data_in[165][6]}} & 8'h1c)^
({{8{data_in[165][7]}} & 8'h38)^
({{8{data_in[166][0]}} & 8'h61)^
({{8{data_in[166][1]}} & 8'hc2)^
({{8{data_in[166][2]}} & 8'haf)^
({{8{data_in[166][3]}} & 8'h75)^
({{8{data_in[166][4]}} & 8'hea)^
({{8{data_in[166][5]}} & 8'hff)^
({{8{data_in[166][6]}} & 8'hd5)^
({{8{data_in[166][7]}} & 8'h81)^
({{8{data_in[167][0]}} & 8'h81)^
({{8{data_in[167][1]}} & 8'h29)^
({{8{data_in[167][2]}} & 8'h52)^
({{8{data_in[167][3]}} & 8'ha4)^
({{8{data_in[167][4]}} & 8'h63)^
({{8{data_in[167][5]}} & 8'hc6)^
({{8{data_in[167][6]}} & 8'ha7)^
({{8{data_in[167][7]}} & 8'h65)^
({{8{data_in[168][0]}} & 8'h6d)^
({{8{data_in[168][1]}} & 8'hda)^
({{8{data_in[168][2]}} & 8'h9f)^
({{8{data_in[168][3]}} & 8'h15)^
({{8{data_in[168][4]}} & 8'h2a)^
({{8{data_in[168][5]}} & 8'h54)^
({{8{data_in[168][6]}} & 8'ha8)^
({{8{data_in[168][7]}} & 8'h7b)^
({{8{data_in[169][0]}} & 8'h69)^
({{8{data_in[169][1]}} & 8'hd2)^
({{8{data_in[169][2]}} & 8'h8f)^
({{8{data_in[169][3]}} & 8'h35)^
({{8{data_in[169][4]}} & 8'h6a)^
({{8{data_in[169][5]}} & 8'hd4)^
({{8{data_in[169][6]}} & 8'h83)^
({{8{data_in[169][7]}} & 8'h2d)^
({{8{data_in[170][0]}} & 8'hc8)^
({{8{data_in[170][1]}} & 8'hbb)^
({{8{data_in[170][2]}} & 8'h5d)^
({{8{data_in[170][3]}} & 8'hba)^
({{8{data_in[170][4]}} & 8'h5f)^
({{8{data_in[170][5]}} & 8'hbe)^
({{8{data_in[170][6]}} & 8'h57)^
({{8{data_in[170][7]}} & 8'hae)^
({{8{data_in[171][0]}} & 8'hd)^
({{8{data_in[171][1]}} & 8'h1a)^
({{8{data_in[171][2]}} & 8'h34)^
({{8{data_in[171][3]}} & 8'h68)^
({{8{data_in[171][4]}} & 8'hd0)^
({{8{data_in[171][5]}} & 8'h8b)^
({{8{data_in[171][6]}} & 8'h3d)^
({{8{data_in[171][7]}} & 8'h7a)^
({{8{data_in[172][0]}} & 8'h89)^
({{8{data_in[172][1]}} & 8'h39)^
({{8{data_in[172][2]}} & 8'h72)^
({{8{data_in[172][3]}} & 8'he4)^
({{8{data_in[172][4]}} & 8'he3)^
({{8{data_in[172][5]}} & 8'hed)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[172][6]}} & 8'hf1)^
({{8{data_in[172][7]}} & 8'hc9)^
({{8{data_in[173][0]}} & 8'hdc)^
({{8{data_in[173][1]}} & 8'h93)^
({{8{data_in[173][2]}} & 8'hd)^
({{8{data_in[173][3]}} & 8'h1a)^
({{8{data_in[173][4]}} & 8'h34)^
({{8{data_in[173][5]}} & 8'h68)^
({{8{data_in[173][6]}} & 8'hd0)^
({{8{data_in[173][7]}} & 8'h8b)^
({{8{data_in[174][0]}} & 8'hfb)^
({{8{data_in[174][1]}} & 8'hdd)^
({{8{data_in[174][2]}} & 8'h91)^
({{8{data_in[174][3]}} & 8'h9)^
({{8{data_in[174][4]}} & 8'h12)^
({{8{data_in[174][5]}} & 8'h24)^
({{8{data_in[174][6]}} & 8'h48)^
({{8{data_in[174][7]}} & 8'h90)^
({{8{data_in[175][0]}} & 8'hc8)^
({{8{data_in[175][1]}} & 8'hbb)^
({{8{data_in[175][2]}} & 8'h5d)^
({{8{data_in[175][3]}} & 8'hba)^
({{8{data_in[175][4]}} & 8'h5f)^
({{8{data_in[175][5]}} & 8'hbe)^
({{8{data_in[175][6]}} & 8'h57)^
({{8{data_in[175][7]}} & 8'hae)^
({{8{data_in[176][0]}} & 8'h30)^
({{8{data_in[176][1]}} & 8'h60)^
({{8{data_in[176][2]}} & 8'hc0)^
({{8{data_in[176][3]}} & 8'hab)^
({{8{data_in[176][4]}} & 8'h7d)^
({{8{data_in[176][5]}} & 8'hfa)^
({{8{data_in[176][6]}} & 8'hdf)^
({{8{data_in[176][7]}} & 8'h95)^
({{8{data_in[177][0]}} & 8'he2)^
({{8{data_in[177][1]}} & 8'hef)^
({{8{data_in[177][2]}} & 8'hf5)^
({{8{data_in[177][3]}} & 8'hc1)^
({{8{data_in[177][4]}} & 8'ha9)^
({{8{data_in[177][5]}} & 8'h79)^
({{8{data_in[177][6]}} & 8'hf2)^
({{8{data_in[177][7]}} & 8'hcf)^
({{8{data_in[178][0]}} & 8'h1b)^
({{8{data_in[178][1]}} & 8'h36)^
({{8{data_in[178][2]}} & 8'h6c)^
({{8{data_in[178][3]}} & 8'hd8)^
({{8{data_in[178][4]}} & 8'h9b)^
({{8{data_in[178][5]}} & 8'h1d)^
({{8{data_in[178][6]}} & 8'h3a)^
({{8{data_in[178][7]}} & 8'h74)^
({{8{data_in[179][0]}} & 8'h8e)^
({{8{data_in[179][1]}} & 8'h37)^
({{8{data_in[179][2]}} & 8'h6e)^
({{8{data_in[179][3]}} & 8'hdc)^
({{8{data_in[179][4]}} & 8'h93)^
({{8{data_in[179][5]}} & 8'hd)^
({{8{data_in[179][6]}} & 8'h1a)^
({{8{data_in[179][7]}} & 8'h34)^
({{8{data_in[180][0]}} & 8'h69)^
({{8{data_in[180][1]}} & 8'hd2)^
({{8{data_in[180][2]}} & 8'h8f)^
({{8{data_in[180][3]}} & 8'h35)^
({{8{data_in[180][4]}} & 8'h6a)^
({{8{data_in[180][5]}} & 8'hd4)^
({{8{data_in[180][6]}} & 8'h83)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[180][7]}} & 8'h2d)^
({{8{data_in[181][0]}} & 8'h33)^
({{8{data_in[181][1]}} & 8'h66)^
({{8{data_in[181][2]}} & 8'hcc)^
({{8{data_in[181][3]}} & 8'hb3)^
({{8{data_in[181][4]}} & 8'h4d)^
({{8{data_in[181][5]}} & 8'h9a)^
({{8{data_in[181][6]}} & 8'h1f)^
({{8{data_in[181][7]}} & 8'h3e)^
({{8{data_in[182][0]}} & 8'hf9)^
({{8{data_in[182][1]}} & 8'hd9)^
({{8{data_in[182][2]}} & 8'h99)^
({{8{data_in[182][3]}} & 8'h19)^
({{8{data_in[182][4]}} & 8'h32)^
({{8{data_in[182][5]}} & 8'h64)^
({{8{data_in[182][6]}} & 8'hc8)^
({{8{data_in[182][7]}} & 8'hbb)^
({{8{data_in[183][0]}} & 8'h8a)^
({{8{data_in[183][1]}} & 8'h3f)^
({{8{data_in[183][2]}} & 8'h7e)^
({{8{data_in[183][3]}} & 8'hfc)^
({{8{data_in[183][4]}} & 8'hd3)^
({{8{data_in[183][5]}} & 8'h8d)^
({{8{data_in[183][6]}} & 8'h31)^
({{8{data_in[183][7]}} & 8'h62)^
({{8{data_in[184][0]}} & 8'h13)^
({{8{data_in[184][1]}} & 8'h26)^
({{8{data_in[184][2]}} & 8'h4c)^
({{8{data_in[184][3]}} & 8'h98)^
({{8{data_in[184][4]}} & 8'h1b)^
({{8{data_in[184][5]}} & 8'h36)^
({{8{data_in[184][6]}} & 8'h6c)^
({{8{data_in[184][7]}} & 8'hd8)^
({{8{data_in[185][0]}} & 8'h22)^
({{8{data_in[185][1]}} & 8'h44)^
({{8{data_in[185][2]}} & 8'h88)^
({{8{data_in[185][3]}} & 8'h3b)^
({{8{data_in[185][4]}} & 8'h76)^
({{8{data_in[185][5]}} & 8'hec)^
({{8{data_in[185][6]}} & 8'hf3)^
({{8{data_in[185][7]}} & 8'hcd)^
({{8{data_in[186][0]}} & 8'hb4)^
({{8{data_in[186][1]}} & 8'h43)^
({{8{data_in[186][2]}} & 8'h86)^
({{8{data_in[186][3]}} & 8'h27)^
({{8{data_in[186][4]}} & 8'h4e)^
({{8{data_in[186][5]}} & 8'h9c)^
({{8{data_in[186][6]}} & 8'h13)^
({{8{data_in[186][7]}} & 8'h26)^
({{8{data_in[187][0]}} & 8'h6)^
({{8{data_in[187][1]}} & 8'hc)^
({{8{data_in[187][2]}} & 8'h18)^
({{8{data_in[187][3]}} & 8'h30)^
({{8{data_in[187][4]}} & 8'h60)^
({{8{data_in[187][5]}} & 8'hc0)^
({{8{data_in[187][6]}} & 8'hab)^
({{8{data_in[187][7]}} & 8'h7d)^
({{8{data_in[188][0]}} & 8'h13)^
({{8{data_in[188][1]}} & 8'h26)^
({{8{data_in[188][2]}} & 8'h4c)^
({{8{data_in[188][3]}} & 8'h98)^
({{8{data_in[188][4]}} & 8'h1b)^
({{8{data_in[188][5]}} & 8'h36)^
({{8{data_in[188][6]}} & 8'h6c)^
({{8{data_in[188][7]}} & 8'hd8)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[189][0]}} & 8'haf)^
({{8{data_in[189][1]}} & 8'h75)^
({{8{data_in[189][2]}} & 8'hea)^
({{8{data_in[189][3]}} & 8'hff)^
({{8{data_in[189][4]}} & 8'hd5)^
({{8{data_in[189][5]}} & 8'h81)^
({{8{data_in[189][6]}} & 8'h29)^
({{8{data_in[189][7]}} & 8'h52)^
({{8{data_in[190][0]}} & 8'he1)^
({{8{data_in[190][1]}} & 8'he9)^
({{8{data_in[190][2]}} & 8'hf9)^
({{8{data_in[190][3]}} & 8'hd9)^
({{8{data_in[190][4]}} & 8'h99)^
({{8{data_in[190][5]}} & 8'h19)^
({{8{data_in[190][6]}} & 8'h32)^
({{8{data_in[190][7]}} & 8'h64)^
({{8{data_in[191][0]}} & 8'hfb)^
({{8{data_in[191][1]}} & 8'hdd)^
({{8{data_in[191][2]}} & 8'h91)^
({{8{data_in[191][3]}} & 8'h9)^
({{8{data_in[191][4]}} & 8'h12)^
({{8{data_in[191][5]}} & 8'h24)^
({{8{data_in[191][6]}} & 8'h48)^
({{8{data_in[191][7]}} & 8'h90)^
({{8{data_in[192][0]}} & 8'h1a)^
({{8{data_in[192][1]}} & 8'h34)^
({{8{data_in[192][2]}} & 8'h68)^
({{8{data_in[192][3]}} & 8'hd0)^
({{8{data_in[192][4]}} & 8'h8b)^
({{8{data_in[192][5]}} & 8'h3d)^
({{8{data_in[192][6]}} & 8'h7a)^
({{8{data_in[192][7]}} & 8'hf4)^
({{8{data_in[193][0]}} & 8'h5c)^
({{8{data_in[193][1]}} & 8'hb8)^
({{8{data_in[193][2]}} & 8'h5b)^
({{8{data_in[193][3]}} & 8'hb6)^
({{8{data_in[193][4]}} & 8'h47)^
({{8{data_in[193][5]}} & 8'h8e)^
({{8{data_in[193][6]}} & 8'h37)^
({{8{data_in[193][7]}} & 8'h6e)^
({{8{data_in[194][0]}} & 8'h6a)^
({{8{data_in[194][1]}} & 8'hd4)^
({{8{data_in[194][2]}} & 8'h83)^
({{8{data_in[194][3]}} & 8'h2d)^
({{8{data_in[194][4]}} & 8'h5a)^
({{8{data_in[194][5]}} & 8'hb4)^
({{8{data_in[194][6]}} & 8'h43)^
({{8{data_in[194][7]}} & 8'h86)^
({{8{data_in[195][0]}} & 8'h10)^
({{8{data_in[195][1]}} & 8'h20)^
({{8{data_in[195][2]}} & 8'h40)^
({{8{data_in[195][3]}} & 8'h80)^
({{8{data_in[195][4]}} & 8'h2b)^
({{8{data_in[195][5]}} & 8'h56)^
({{8{data_in[195][6]}} & 8'hac)^
({{8{data_in[195][7]}} & 8'h73)^
({{8{data_in[196][0]}} & 8'hc0)^
({{8{data_in[196][1]}} & 8'hab)^
({{8{data_in[196][2]}} & 8'h7d)^
({{8{data_in[196][3]}} & 8'hfa)^
({{8{data_in[196][4]}} & 8'hdf)^
({{8{data_in[196][5]}} & 8'h95)^
({{8{data_in[196][6]}} & 8'h1)^
({{8{data_in[196][7]}} & 8'h2)^
({{8{data_in[197][0]}} & 8'h96)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[197][1]}} & 8'h7)^
({{8{data_in[197][2]}} & 8'he)^
({{8{data_in[197][3]}} & 8'h1c)^
({{8{data_in[197][4]}} & 8'h38)^
({{8{data_in[197][5]}} & 8'h70)^
({{8{data_in[197][6]}} & 8'he0)^
({{8{data_in[197][7]}} & 8'heb)^
({{8{data_in[198][0]}} & 8'h1a)^
({{8{data_in[198][1]}} & 8'h34)^
({{8{data_in[198][2]}} & 8'h68)^
({{8{data_in[198][3]}} & 8'hd0)^
({{8{data_in[198][4]}} & 8'h8b)^
({{8{data_in[198][5]}} & 8'h3d)^
({{8{data_in[198][6]}} & 8'h7a)^
({{8{data_in[198][7]}} & 8'hf4)^
({{8{data_in[199][0]}} & 8'ha0)^
({{8{data_in[199][1]}} & 8'h6b)^
({{8{data_in[199][2]}} & 8'hd6)^
({{8{data_in[199][3]}} & 8'h87)^
({{8{data_in[199][4]}} & 8'h25)^
({{8{data_in[199][5]}} & 8'h4a)^
({{8{data_in[199][6]}} & 8'h94)^
({{8{data_in[199][7]}} & 8'h3)^
({{8{data_in[200][0]}} & 8'he9)^
({{8{data_in[200][1]}} & 8'hf9)^
({{8{data_in[200][2]}} & 8'hd9)^
({{8{data_in[200][3]}} & 8'h99)^
({{8{data_in[200][4]}} & 8'h19)^
({{8{data_in[200][5]}} & 8'h32)^
({{8{data_in[200][6]}} & 8'h64)^
({{8{data_in[200][7]}} & 8'hc8)^
({{8{data_in[201][0]}} & 8'h82)^
({{8{data_in[201][1]}} & 8'h2f)^
({{8{data_in[201][2]}} & 8'h5e)^
({{8{data_in[201][3]}} & 8'hbc)^
({{8{data_in[201][4]}} & 8'h53)^
({{8{data_in[201][5]}} & 8'ha6)^
({{8{data_in[201][6]}} & 8'h67)^
({{8{data_in[201][7]}} & 8'hce)^
({{8{data_in[202][0]}} & 8'hcf)^
({{8{data_in[202][1]}} & 8'hb5)^
({{8{data_in[202][2]}} & 8'h41)^
({{8{data_in[202][3]}} & 8'h82)^
({{8{data_in[202][4]}} & 8'h2f)^
({{8{data_in[202][5]}} & 8'h5e)^
({{8{data_in[202][6]}} & 8'hbc)^
({{8{data_in[202][7]}} & 8'h53)^
({{8{data_in[203][0]}} & 8'h36)^
({{8{data_in[203][1]}} & 8'h6c)^
({{8{data_in[203][2]}} & 8'hd8)^
({{8{data_in[203][3]}} & 8'h9b)^
({{8{data_in[203][4]}} & 8'h1d)^
({{8{data_in[203][5]}} & 8'h3a)^
({{8{data_in[203][6]}} & 8'h74)^
({{8{data_in[203][7]}} & 8'he8)^
({{8{data_in[204][0]}} & 8'hae)^
({{8{data_in[204][1]}} & 8'h77)^
({{8{data_in[204][2]}} & 8'hee)^
({{8{data_in[204][3]}} & 8'hf7)^
({{8{data_in[204][4]}} & 8'hc5)^
({{8{data_in[204][5]}} & 8'ha1)^
({{8{data_in[204][6]}} & 8'h69)^
({{8{data_in[204][7]}} & 8'hd2)^
({{8{data_in[205][0]}} & 8'h16)^
({{8{data_in[205][1]}} & 8'h2c)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[205][2]}} & 8'h58)^
({{8{data_in[205][3]}} & 8'hb0)^
({{8{data_in[205][4]}} & 8'h4b)^
({{8{data_in[205][5]}} & 8'h96)^
({{8{data_in[205][6]}} & 8'h7)^
({{8{data_in[205][7]}} & 8'he)^
({{8{data_in[206][0]}} & 8'h1a)^
({{8{data_in[206][1]}} & 8'h34)^
({{8{data_in[206][2]}} & 8'h68)^
({{8{data_in[206][3]}} & 8'hd0)^
({{8{data_in[206][4]}} & 8'h8b)^
({{8{data_in[206][5]}} & 8'h3d)^
({{8{data_in[206][6]}} & 8'h7a)^
({{8{data_in[206][7]}} & 8'hf4)^
({{8{data_in[207][0]}} & 8'hf3)^
({{8{data_in[207][1]}} & 8'hcd)^
({{8{data_in[207][2]}} & 8'hb1)^
({{8{data_in[207][3]}} & 8'h49)^
({{8{data_in[207][4]}} & 8'h92)^
({{8{data_in[207][5]}} & 8'hf)^
({{8{data_in[207][6]}} & 8'h1e)^
({{8{data_in[207][7]}} & 8'h3c)^
({{8{data_in[208][0]}} & 8'h8c)^
({{8{data_in[208][1]}} & 8'h33)^
({{8{data_in[208][2]}} & 8'h66)^
({{8{data_in[208][3]}} & 8'hcc)^
({{8{data_in[208][4]}} & 8'hb3)^
({{8{data_in[208][5]}} & 8'h4d)^
({{8{data_in[208][6]}} & 8'h9a)^
({{8{data_in[208][7]}} & 8'h1f)^
({{8{data_in[209][0]}} & 8'h3d)^
({{8{data_in[209][1]}} & 8'h7a)^
({{8{data_in[209][2]}} & 8'hf4)^
({{8{data_in[209][3]}} & 8'hc3)^
({{8{data_in[209][4]}} & 8'had)^
({{8{data_in[209][5]}} & 8'h71)^
({{8{data_in[209][6]}} & 8'he2)^
({{8{data_in[209][7]}} & 8'hef)^
({{8{data_in[210][0]}} & 8'ha9)^
({{8{data_in[210][1]}} & 8'h79)^
({{8{data_in[210][2]}} & 8'hf2)^
({{8{data_in[210][3]}} & 8'hcf)^
({{8{data_in[210][4]}} & 8'hb5)^
({{8{data_in[210][5]}} & 8'h41)^
({{8{data_in[210][6]}} & 8'h82)^
({{8{data_in[210][7]}} & 8'h2f)^
({{8{data_in[211][0]}} & 8'ha)^
({{8{data_in[211][1]}} & 8'h14)^
({{8{data_in[211][2]}} & 8'h28)^
({{8{data_in[211][3]}} & 8'h50)^
({{8{data_in[211][4]}} & 8'ha0)^
({{8{data_in[211][5]}} & 8'h6b)^
({{8{data_in[211][6]}} & 8'hd6)^
({{8{data_in[211][7]}} & 8'h87)^
({{8{data_in[212][0]}} & 8'hab)^
({{8{data_in[212][1]}} & 8'h7d)^
({{8{data_in[212][2]}} & 8'hfa)^
({{8{data_in[212][3]}} & 8'hdf)^
({{8{data_in[212][4]}} & 8'h95)^
({{8{data_in[212][5]}} & 8'h1)^
({{8{data_in[212][6]}} & 8'h2)^
({{8{data_in[212][7]}} & 8'h4)^
({{8{data_in[213][0]}} & 8'hce)^
({{8{data_in[213][1]}} & 8'hb7)^
({{8{data_in[213][2]}} & 8'h45)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[213][3]}} & 8'h8a)^
({{8{data_in[213][4]}} & 8'h3f)^
({{8{data_in[213][5]}} & 8'h7e)^
({{8{data_in[213][6]}} & 8'hfc)^
({{8{data_in[213][7]}} & 8'hd3)^
({{8{data_in[214][0]}} & 8'hd1)^
({{8{data_in[214][1]}} & 8'h89)^
({{8{data_in[214][2]}} & 8'h39)^
({{8{data_in[214][3]}} & 8'h72)^
({{8{data_in[214][4]}} & 8'he4)^
({{8{data_in[214][5]}} & 8'he3)^
({{8{data_in[214][6]}} & 8'hed)^
({{8{data_in[214][7]}} & 8'hf1)^
({{8{data_in[215][0]}} & 8'h48)^
({{8{data_in[215][1]}} & 8'h90)^
({{8{data_in[215][2]}} & 8'hb)^
({{8{data_in[215][3]}} & 8'h16)^
({{8{data_in[215][4]}} & 8'h2c)^
({{8{data_in[215][5]}} & 8'h58)^
({{8{data_in[215][6]}} & 8'hb0)^
({{8{data_in[215][7]}} & 8'h4b)^
({{8{data_in[216][0]}} & 8'h7f)^
({{8{data_in[216][1]}} & 8'hfe)^
({{8{data_in[216][2]}} & 8'hd7)^
({{8{data_in[216][3]}} & 8'h85)^
({{8{data_in[216][4]}} & 8'h21)^
({{8{data_in[216][5]}} & 8'h42)^
({{8{data_in[216][6]}} & 8'h84)^
({{8{data_in[216][7]}} & 8'h23)^
({{8{data_in[217][0]}} & 8'ha8)^
({{8{data_in[217][1]}} & 8'h7b)^
({{8{data_in[217][2]}} & 8'hf6)^
({{8{data_in[217][3]}} & 8'hc7)^
({{8{data_in[217][4]}} & 8'ha5)^
({{8{data_in[217][5]}} & 8'h61)^
({{8{data_in[217][6]}} & 8'hc2)^
({{8{data_in[217][7]}} & 8'haf)^
({{8{data_in[218][0]}} & 8'hde)^
({{8{data_in[218][1]}} & 8'h97)^
({{8{data_in[218][2]}} & 8'h5)^
({{8{data_in[218][3]}} & 8'ha)^
({{8{data_in[218][4]}} & 8'h14)^
({{8{data_in[218][5]}} & 8'h28)^
({{8{data_in[218][6]}} & 8'h50)^
({{8{data_in[218][7]}} & 8'ha0)^
({{8{data_in[219][0]}} & 8'h6d)^
({{8{data_in[219][1]}} & 8'hda)^
({{8{data_in[219][2]}} & 8'h9f)^
({{8{data_in[219][3]}} & 8'h15)^
({{8{data_in[219][4]}} & 8'h2a)^
({{8{data_in[219][5]}} & 8'h54)^
({{8{data_in[219][6]}} & 8'ha8)^
({{8{data_in[219][7]}} & 8'h7b)^
({{8{data_in[220][0]}} & 8'h48)^
({{8{data_in[220][1]}} & 8'h90)^
({{8{data_in[220][2]}} & 8'hb)^
({{8{data_in[220][3]}} & 8'h16)^
({{8{data_in[220][4]}} & 8'h2c)^
({{8{data_in[220][5]}} & 8'h58)^
({{8{data_in[220][6]}} & 8'hb0)^
({{8{data_in[220][7]}} & 8'h4b)^
({{8{data_in[221][0]}} & 8'h38)^
({{8{data_in[221][1]}} & 8'h70)^
({{8{data_in[221][2]}} & 8'he0)^
({{8{data_in[221][3]}} & 8'heb)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[221][4]}} & 8'hfd)^
({{8{data_in[221][5]}} & 8'hd1)^
({{8{data_in[221][6]}} & 8'h89)^
({{8{data_in[221][7]}} & 8'h39)^
({{8{data_in[222][0]}} & 8'h42)^
({{8{data_in[222][1]}} & 8'h84)^
({{8{data_in[222][2]}} & 8'h23)^
({{8{data_in[222][3]}} & 8'h46)^
({{8{data_in[222][4]}} & 8'h8c)^
({{8{data_in[222][5]}} & 8'h33)^
({{8{data_in[222][6]}} & 8'h66)^
({{8{data_in[222][7]}} & 8'hcc)^
({{8{data_in[223][0]}} & 8'hf)^
({{8{data_in[223][1]}} & 8'h1e)^
({{8{data_in[223][2]}} & 8'h3c)^
({{8{data_in[223][3]}} & 8'h78)^
({{8{data_in[223][4]}} & 8'hf0)^
({{8{data_in[223][5]}} & 8'hcb)^
({{8{data_in[223][6]}} & 8'hbd)^
({{8{data_in[223][7]}} & 8'h51)^
({{8{data_in[224][0]}} & 8'h81)^
({{8{data_in[224][1]}} & 8'h29)^
({{8{data_in[224][2]}} & 8'h52)^
({{8{data_in[224][3]}} & 8'ha4)^
({{8{data_in[224][4]}} & 8'h63)^
({{8{data_in[224][5]}} & 8'hc6)^
({{8{data_in[224][6]}} & 8'ha7)^
({{8{data_in[224][7]}} & 8'h65)^
({{8{data_in[225][0]}} & 8'hc9)^
({{8{data_in[225][1]}} & 8'hb9)^
({{8{data_in[225][2]}} & 8'h59)^
({{8{data_in[225][3]}} & 8'hb2)^
({{8{data_in[225][4]}} & 8'h4f)^
({{8{data_in[225][5]}} & 8'h9e)^
({{8{data_in[225][6]}} & 8'h17)^
({{8{data_in[225][7]}} & 8'h2e)^
({{8{data_in[226][0]}} & 8'h74)^
({{8{data_in[226][1]}} & 8'he8)^
({{8{data_in[226][2]}} & 8'hfb)^
({{8{data_in[226][3]}} & 8'hdd)^
({{8{data_in[226][4]}} & 8'h91)^
({{8{data_in[226][5]}} & 8'h9)^
({{8{data_in[226][6]}} & 8'h12)^
({{8{data_in[226][7]}} & 8'h24)^
({{8{data_in[227][0]}} & 8'h1e)^
({{8{data_in[227][1]}} & 8'h3c)^
({{8{data_in[227][2]}} & 8'h78)^
({{8{data_in[227][3]}} & 8'hf0)^
({{8{data_in[227][4]}} & 8'hcb)^
({{8{data_in[227][5]}} & 8'hbd)^
({{8{data_in[227][6]}} & 8'h51)^
({{8{data_in[227][7]}} & 8'ha2)^
({{8{data_in[228][0]}} & 8'h34)^
({{8{data_in[228][1]}} & 8'h68)^
({{8{data_in[228][2]}} & 8'hd0)^
({{8{data_in[228][3]}} & 8'h8b)^
({{8{data_in[228][4]}} & 8'h3d)^
({{8{data_in[228][5]}} & 8'h7a)^
({{8{data_in[228][6]}} & 8'hf4)^
({{8{data_in[228][7]}} & 8'hc3)^
({{8{data_in[229][0]}} & 8'h82)^
({{8{data_in[229][1]}} & 8'h2f)^
({{8{data_in[229][2]}} & 8'h5e)^
({{8{data_in[229][3]}} & 8'hbc)^
({{8{data_in[229][4]}} & 8'h53)^

```

Base 6.4 vs Base 6.3

```

({{8{data_in[229][5]}} & 8'ha6)^
({{8{data_in[229][6]}} & 8'h67)^
({{8{data_in[229][7]}} & 8'hce)^
({{8{data_in[230][0]}} & 8'hed)^
({{8{data_in[230][1]}} & 8'hf1)^
({{8{data_in[230][2]}} & 8'hc9)^
({{8{data_in[230][3]}} & 8'hb9)^
({{8{data_in[230][4]}} & 8'h59)^
({{8{data_in[230][5]}} & 8'hb2)^
({{8{data_in[230][6]}} & 8'h4f)^
({{8{data_in[230][7]}} & 8'h9e)^
({{8{data_in[231][0]}} & 8'h71)^
({{8{data_in[231][1]}} & 8'he2)^
({{8{data_in[231][2]}} & 8'hef)^
({{8{data_in[231][3]}} & 8'hf5)^
({{8{data_in[231][4]}} & 8'hc1)^
({{8{data_in[231][5]}} & 8'ha9)^
({{8{data_in[231][6]}} & 8'h79)^
({{8{data_in[231][7]}} & 8'hf2)^
({{8{data_in[232][0]}} & 8'h83)^
({{8{data_in[232][1]}} & 8'h2d)^
({{8{data_in[232][2]}} & 8'h5a)^
({{8{data_in[232][3]}} & 8'hb4)^
({{8{data_in[232][4]}} & 8'h43)^
({{8{data_in[232][5]}} & 8'h86)^
({{8{data_in[232][6]}} & 8'h27)^
({{8{data_in[232][7]}} & 8'h4e)^
({{8{data_in[233][0]}} & 8'h80)^
({{8{data_in[233][1]}} & 8'h2b)^
({{8{data_in[233][2]}} & 8'h56)^
({{8{data_in[233][3]}} & 8'hac)^
({{8{data_in[233][4]}} & 8'h73)^
({{8{data_in[233][5]}} & 8'he6)^
({{8{data_in[233][6]}} & 8'he7)^
({{8{data_in[233][7]}} & 8'he5)^
({{8{data_in[234][0]}} & 8'h1c)^
({{8{data_in[234][1]}} & 8'h38)^
({{8{data_in[234][2]}} & 8'h70)^
({{8{data_in[234][3]}} & 8'he0)^
({{8{data_in[234][4]}} & 8'heb)^
({{8{data_in[234][5]}} & 8'hfd)^
({{8{data_in[234][6]}} & 8'hd1)^
({{8{data_in[234][7]}} & 8'h89)^
({{8{data_in[235][0]}} & 8'h7b)^
({{8{data_in[235][1]}} & 8'hf6)^
({{8{data_in[235][2]}} & 8'hc7)^
({{8{data_in[235][3]}} & 8'ha5)^
({{8{data_in[235][4]}} & 8'h61)^
({{8{data_in[235][5]}} & 8'hc2)^
({{8{data_in[235][6]}} & 8'haf)^
({{8{data_in[235][7]}} & 8'h75)^
({{8{data_in[236][0]}} & 8'hbe)^
({{8{data_in[236][1]}} & 8'h57)^
({{8{data_in[236][2]}} & 8'hae)^
({{8{data_in[236][3]}} & 8'h77)^
({{8{data_in[236][4]}} & 8'hee)^
({{8{data_in[236][5]}} & 8'hf7)^
({{8{data_in[236][6]}} & 8'hc5)^
({{8{data_in[236][7]}} & 8'ha1)^
({{8{data_in[237][0]}} & 8'h7c)^
({{8{data_in[237][1]}} & 8'hf8)^
({{8{data_in[237][2]}} & 8'hdb)^
({{8{data_in[237][3]}} & 8'h9d)^
({{8{data_in[237][4]}} & 8'h11)^
({{8{data_in[237][5]}} & 8'h22)^

```

```
{ {8{data_in[237][6]}} & 8'h44)^  
({8{data_in[237][7]}} & 8'h88)^  
({8{data_in[238][0]}} & 8'hfb)^  
({8{data_in[238][1]}} & 8'hdd)^  
({8{data_in[238][2]}} & 8'h91)^  
({8{data_in[238][3]}} & 8'h9)^  
({8{data_in[238][4]}} & 8'h12)^  
({8{data_in[238][5]}} & 8'h24)^  
({8{data_in[238][6]}} & 8'h48)^  
({8{data_in[238][7]}} & 8'h90)^  
({8{data_in[239][0]}} & 8'h80)^  
({8{data_in[239][1]}} & 8'h2b)^  
({8{data_in[239][2]}} & 8'h56)^  
({8{data_in[239][3]}} & 8'hac)^  
({8{data_in[239][4]}} & 8'h73)^  
({8{data_in[239][5]}} & 8'he6)^  
({8{data_in[239][6]}} & 8'he7)^  
({8{data_in[239][7]}} & 8'he5)^  
({8{data_in[240][0]}} & 8'h22)^  
({8{data_in[240][1]}} & 8'h44)^  
({8{data_in[240][2]}} & 8'h88)^  
({8{data_in[240][3]}} & 8'h3b)^  
({8{data_in[240][4]}} & 8'h76)^  
({8{data_in[240][5]}} & 8'hec)^  
({8{data_in[240][6]}} & 8'hf3)^  
({8{data_in[240][7]}} & 8'hcd)^  
({8{data_in[241][0]}} & 8'h69)^  
({8{data_in[241][1]}} & 8'hd2)^  
({8{data_in[241][2]}} & 8'h8f)^  
({8{data_in[241][3]}} & 8'h35)^  
({8{data_in[241][4]}} & 8'h6a)^  
({8{data_in[241][5]}} & 8'hd4)^  
({8{data_in[241][6]}} & 8'h83)^  
({8{data_in[241][7]}} & 8'h2d);  
end  
endmodule  
'endif
```

L. Example IDE TLPs and Test Keys §

This appendix includes several example IDE TLPs, intended to be used to verify correct implementation of IDE. Also included are a set of keys to be used for testing, to simplify IDE validation.

L.1 Example NFM IDE TLP Without Partial Header Encryption §

The TLP shown in § Figure L-1 , when processed using the Key and IV shown in § Table L-1 , which also includes other relevant inputs and outputs, yields the IDE TLP shown in § Figure L-2 .

The diagram illustrates the structure of a memory copy operation across 10 bytes. The fields and their bit ranges are as follows:

- Byte 0:**
 - +0: Fmt (bits 7-0), Type (bits 5-0), T9 (bit 4), TC (bit 3), T8 (bit 2), A2 (bit 1), R (bit 0).
 - +1: TH (bit 7), TD (bit 6), EP (bit 5), Attr (bit 4), AT (bit 3), Length (bits 2-0).
 - +2: Last DW BE (bits 7-0).
 - +3: First DW BE (bits 7-0).
- Byte 4:** Requester ID (bits 15-0).
- Byte 8:** Address[63:32] (bits 63-0).
- Byte 12:** Address[31:2] (bits 31-0).
- Byte 16:** DATA DW0 (bits 31-0).
- Byte 20:** DATA DW1 (bits 31-0).
- Byte 24:** DATA DW2 (bits 31-0).
- Byte 28:** DATA DW3 (bits 31-0).

Figure L-1 Example Memory Write TLP (NFM) §

Table L-1 Inputs and Outputs for Example IDE TLP

| | |
|--|--|
| Input/Outputs variables for AES GCM | Values Byte0 [7:0] byte 1 [15:8] Byte N Key, IV, MAC : Follows AES-GCM Specification with Most significant leftmost |
|--|--|

Inputs

| | |
|----------------------------|--|
| Key (K) | 8A 95 85 8A 23 1D 41 4A 37 33 F0 E5 7A F6 AE 77 C7 F0 CF BA 75 54 D6 70 83 34 2A 64 2A EF 9D 5D |
| Initialization Vector (IV) | 00 00 00 00 00 00 00 00 00 00 00 00 01 |
| A | 92 00 BC 0C 60 00 00 04 AB CD 00 FF 00 00 00 01 00 00 00 00 |
| P (including PCRC) | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 88 E2 CE CE |

Base 6.4 VS Base 6.3

| | |
|-------------------------------------|--|
| Input/Outputs variables for AES GCM | Values Byte0 [7:0] byte 1 [15:8] Byte N Key, IV, MAC : Follows AES-GCM Specification with Most significant leftmost |
|-------------------------------------|--|

Outputs

| | |
|------------------------------|---|
| C (including encrypted PCRC) | 0C F7 B4 8F E6 30 D7 18 CE B9 F4 23 DE EB 68 53 7C DF 22 EE |
| MAC | 53 A7 D7 13 7F 48 1E 7E FB 97 E8 C6 |

```
{} , { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 416, "isMessage": true, "defaultUnused": "R", "fields": [{"lsbyte": 0, "lsbit": 5, "msbyte": 0, "msbit": 7, "value": ["1", "0", "0"], "name": "Fmt", "attr": "ro"}, {"lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 4, "value": ["1", "0", "0", "1", "0"], "attr": "ro"}, {"lsbyte": 1, "lsbit": 0, "msbyte": 1, "msbit": 7, "value": ["0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 2, "lsbit": 0, "msbyte": 2, "msbit": 7, "value": ["1", "0", "1", "1", "1", "0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 3, "msbyte": 3, "msbit": 6, "value": ["0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 2, "msbyte": 2, "msbit": 4, "value": ["1"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 1, "msbyte": 3, "msbit": 1, "value": ["0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 0, "msbit": 5, "value": ["0"], "attr": "ro"}, {"lsbyte": 4, "lsbit": 4, "msbyte": 4, "msbit": 4, "value": ["0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 7, "msbyte": 5, "msbit": 7, "value": ["0"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 6, "msbyte": 5, "msbit": 4, "value": ["TC"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 5, "msbyte": 5, "msbit": 3, "value": ["T8"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 4, "msbyte": 5, "msbit": 2, "value": ["A2"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 3, "msbyte": 5, "msbit": 2, "value": ["TD"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 2, "msbyte": 5, "msbit": 5, "value": ["Attr"], "attr": "ro"}, {"lsbyte": 6, "lsbit": 6, "msbyte": 6, "msbit": 6, "value": ["EP"], "attr": "ro"}, {"lsbyte": 6, "lsbit": 5, "msbyte": 6, "msbit": 4, "value": ["AT"], "attr": "ro"}, {"lsbyte": 6, "lsbit": 4, "msbyte": 6, "msbit": 1, "value": ["Length"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 0, "msbit": 1, "value": ["Requester ID"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 0, "msbit": 10, "value": ["Last DW BE"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 11, "msbyte": 10, "msbit": 3, "value": ["First DW BE"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 15, "msbyte": 0, "msbit": 12, "value": ["Address[63:32]"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 19, "msbyte": 19, "msbit": 7, "value": ["Address[31:2]"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 27, "msbit": 24, "value": ["DATA DW1"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 28, "msbit": 27, "value": ["DATA DW2"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 35, "msbit": 32, "value": ["DATA DW3"], "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 39, "msbit": 36, "value": ["PCRC"], "attr": "ro"}]
```

Base 6.4 vs Base 6.3

```
"addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 43, "lsbit": 0, "msbyte": 40, "msbit": 7, "name": "MAC DW0",  
"value": ["0","0","1","0","1","0","0","1","1","1","0","1","0","0","1","1","1","1","0","1","0","1","1","1","0","0","1","0","0","1","1","1"],  
"addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 47, "lsbit": 0, "msbyte": 44, "msbit": 7, "name": "MAC DW1",  
"value": ["0","1","1","1","1","1","1","1","0","1","0","0","1","0","0","0","1","1","1","1","0","1","0","1","1","1","1","1","0"],  
"addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 51, "lsbit": 0, "msbyte": 48, "msbit": 7, "name": "MAC DW2",  
"value": ["1","1","1","1","1","1","0","1","1","1","1","0","0","1","1","1","1","1","1","0","1","0","0","0","1","1","0","0","0","1","1","0"],  
"addClass": "regFieldSmallText", "attr": "ro"}]}]
```

Figure L-2 Example NFM Memory Write IDE TLP §

L.2 Example FM IDE TLP Without Partial Header Encryption §

The TLP shown in § Figure L-3 , when processed using the Key and IV shown in § Table L-2 , which also includes other relevant inputs and outputs, yields the IDE TLP shown in § Figure L-4 .

```
], { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "value": [ "0", "1", "1", "0", "0", "0", "0" ], "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 5, "name": "TC", "value": [ "0", "0", "0" ], "attr": "ro" }, { "lsbyte": 1, "lsbit": 4, "msbyte": 1, "msbit": 0, "name": "OHC", "value": [ "0", "0", "1", "0", "1" ], "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 5, "name": "TS", "value": [ "0", "0", "0" ], "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 2, "name": "Attr", "value": [ "0", "0", "0" ], "attr": "ro" }, { "lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": [ "0", "0", "0", "0", "0", "1", "0", "0" ], "attr": "ro" }, { "lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "RequesterID", "value": [ "1", "0", "1", "0", "1", "0", "1", "1", "1", "0", "0", "1", "1", "0", "1", "1" ], "attr": "ro" }, { "lsbyte": 6, "lsbit": 7,
```

Base 6.4 vs Base 6.3

```

"msbyte": 6, "msbit": 7, "name": "EP", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0,
"msbyte": 6, "msbit": 5, "name": "Tag", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[31:2]", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 15, "lsbit": 2, "msbyte": 12, "msbit": 7, "name": "Address[1:0]", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "AT", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 16, "lsbit": 7, "msbyte": 16, "msbit": 7, "name": "NW", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 16, "lsbit": 6, "msbyte": 16, "msbit": 6, "name": "PV", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 16, "lsbit": 5, "msbyte": 16, "msbit": 5, "name": "PMR", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 16, "lsbit": 4, "msbyte": 16, "msbit": 4, "name": "ER", "value": ["0", "0"], "attr": "ro"}, {"lsbyte": 16, "lsbit": 3, "msbyte": 18, "msbit": 0, "name": "PASID", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 19, "lsbit": 4, "msbyte": 19, "msbit": 7, "name": "Last DW BE", "value": ["1", "1", "1", "1"], "attr": "ro"}, {"lsbyte": 19, "lsbit": 0, "msbyte": 19, "msbit": 3, "name": "First DW BE", "value": ["1", "1", "1", "1"], "attr": "ro"}, {"lsbyte": 20, "lsbit": 0, "msbyte": 20, "msbit": 7, "name": "Requester Segment", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 21, "lsbit": 0, "msbyte": 21, "msbit": 7, "name": "PR_Sent_Counter", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 22, "lsbit": 0, "msbyte": 22, "msbit": 7, "name": "Stream_ID", "value": ["0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 23, "lsbit": 4, "msbyte": 23, "msbit": 6, "name": "Sub Stream", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 23, "lsbit": 3, "msbyte": 23, "msbit": 3, "name": "RSV", "value": ["0", "0", "0"], "attr": "ro"}, {"lsbyte": 23, "lsbit": 1, "msbyte": 23, "msbit": 1, "name": "K", "value": ["1"], "attr": "ro"}, {"lsbyte": 23, "lsbit": 0, "msbyte": 23, "msbit": 0, "name": "T", "value": ["0"], "attr": "ro"}, {"lsbyte": 27, "lsbit": 0, "msbyte": 24, "msbit": 7, "name": "DATA DWO", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 31, "lsbit": 0, "msbyte": 28, "msbit": 7, "name": "DATA DW1", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 35, "lsbit": 0, "msbyte": 32, "msbit": 7, "name": "DATA DW2", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 39, "lsbit": 0, "msbyte": 36, "msbit": 7, "name": "DATA DW3", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}]]]

```

Base 6.4 vs Base 6.3

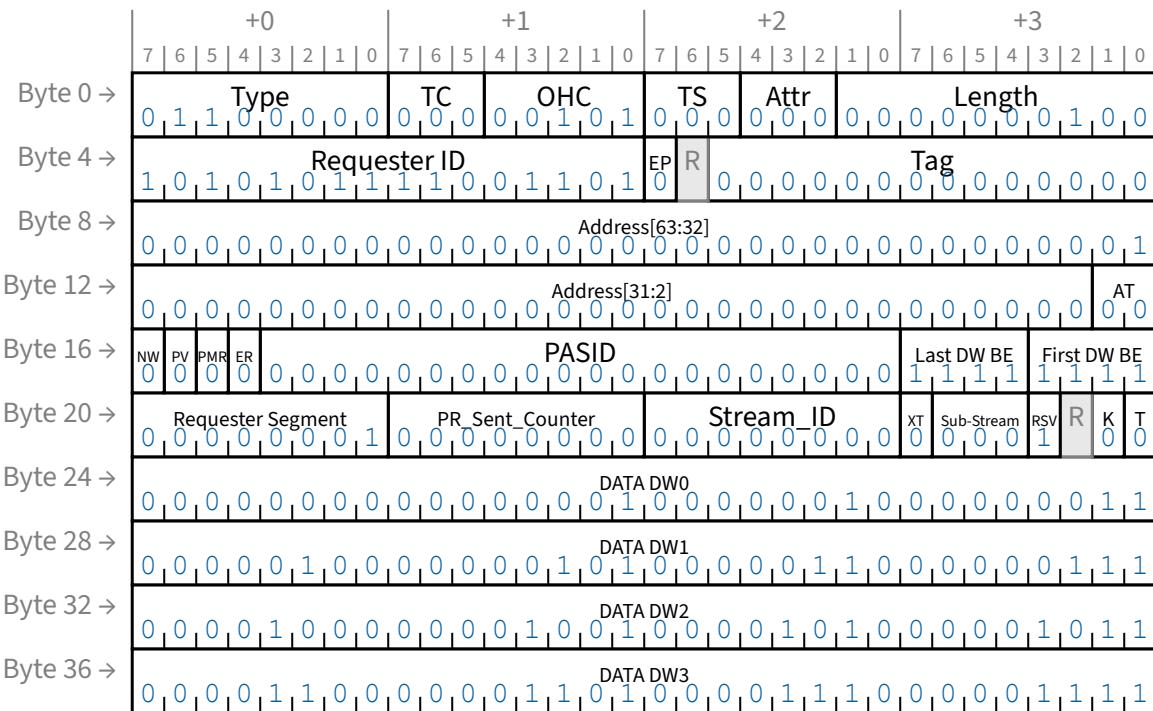


Figure L-3 Example Memory Write TLP (FM) §

Table L-2 Inputs and Outputs for Example IDE TLP (FM) §

| | |
|-------------------------------------|--|
| Input/Outputs variables for AES GCM | Values Byte0 [7:0] byte 1 [15:8] Byte N Key, IV, MAC : Follows AES-GCM Specification with Most significant leftmost |
|-------------------------------------|--|

Inputs

| | |
|----------------------------|--|
| Key (K) | 8A 95 85 8A 23 1D 41 4A 37 33 F0 E5 7A F6 AE 77 C7 F0 CF BA 75 54 D6 70 83 34 2A 64 2A EF 9D 5D |
| Initialization Vector (IV) | 00 00 00 00 00 00 00 00 00 00 00 00 00 01 |
| A | 60 05 C0 04 AB CD 00 00 00 00 01 00 00 00 00 00 00 00 FF 01 00 BC 08 |
| P (including PCRC) | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 88 E2 CE CE |

Outputs

| | |
|------------------------------|---|
| C (including encrypted PCRC) | 0C F7 B4 8F E6 30 D7 18 CE B9 F4 23 DE EB 68 53 7C DF 22 EE |
| MAC | D8 DC 27 8B EF 37 7B 40 D5 1A D6 28 |

```
], { "debug": false, "preClass": "hide", "cellwidth": 15, "width": 448, "isMessage": true, "defaultUnused": "R", "fields": [ { "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "value": [ "0", "1", "1", "0", "0", "0", "0", "0" ], "name": "Type", "attr": "ro" }, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 5, "name": "TC", "value": [ "0", "0", "0" ], "attr": "ro" }, { "lsbyte": 1, "lsbit": 4, "msbyte": 1, "msbit": 0, "name": "OHC", "value": [ "0", "0", "1", "0", "1" ], "attr": "ro" }, { "lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 5, "name": "TS", "value": [ "1", "1", "0" ], "attr": "ro" }, { "lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 2, "name": "Pulse", "value": [ "0", "0", "0", "0" ], "attr": "ro" } ] } ] }
```

Base 6.4 vs Base 6.3

"Attr", "value": ["0", "0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "0", "1", "0", "0"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "value": ["1", "0", "1", "0", "1", "0", "1", "1", "1", "0", "0", "1", "1", "0", "1"], "attr": "ro"}, {"lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 7, "name": "EP", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 5, "name": "Tag", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 15, "lsbit": 2, "msbyte": 12, "msbit": 7, "name": "Address[31:2]", "value": ["0", "0"], "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 15, "lsbit": 2, "msbyte": 12, "msbit": 7, "name": "Address[31:2]", "value": ["0", "0"], "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "AT", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 7, "msbyte": 16, "msbit": 7, "name": "NW", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 6, "msbyte": 16, "msbit": 6, "name": "PV", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 5, "msbyte": 16, "msbit": 5, "name": "PMR", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 4, "msbyte": 16, "msbit": 4, "name": "ER", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 3, "msbyte": 18, "msbit": 0, "name": "PASID", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 19, "lsbit": 4, "msbyte": 19, "msbit": 7, "name": "Last DW BE", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 19, "lsbit": 0, "msbyte": 19, "msbit": 3, "name": "First DW BE", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 20, "lsbit": 0, "msbyte": 20, "msbit": 7, "name": "Requester Segment", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 21, "lsbit": 0, "msbyte": 21, "msbit": 7, "name": "PR_Sent_Counter", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 22, "lsbit": 0, "msbyte": 22, "msbit": 7, "name": "Stream_ID", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 3, "msbyte": 23, "msbit": 3, "name": "RSV", "value": "1", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 1, "msbyte": 23, "msbit": 1, "name": "K", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 0, "msbyte": 23, "msbit": 0, "name": "T", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 27, "lsbit": 0, "msbyte": 24, "msbit": 7, "name": "DATA DW0", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 31, "lsbit": 0, "msbyte": 28, "msbit": 7, "name": "DATA DW1", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 35, "lsbit": 0, "msbyte": 32, "msbit": 7, "name": "DATA DW2", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 39, "lsbit": 0, "msbyte": 36, "msbit": 7, "name": "DATA DW3", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 43, "lsbit": 0, "msbyte": 40, "msbit": 7, "name": "PCRC", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 47, "lsbit": 0, "msbyte": 44, "msbit": 7, "name": "MAC DW0", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 51, "lsbit": 0, "msbyte": 48, "msbit": 7, "name": "MAC DW1", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 55, "lsbit": 0, "msbyte": 52, "msbit": 7, "name": "MAC DW2", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}]]

Base 6.4 vs Base 6.3

| | +0 | +1 | +2 | +3 |
|-----------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| Byte 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 | 7 6 5 4 3 2 1 0 |
| | Type | TC | OHC | TS Attr Length |
| Byte 4 → | 1 0 1 0 1 0 1 1 | 1 1 0 0 1 1 0 1 | EP R | Tag |
| Byte 8 → | 0 0 0 0 0 0 0 0 | Address[63:32] | | |
| Byte 12 → | 0 0 0 0 0 0 0 0 | Address[31:2] | | |
| Byte 16 → | NW PV PMR ER | PASID | Last DW BE | First DW BE |
| Byte 20 → | Requester Segment | PR_Sent_Counter | Stream_ID | XT Sub-Stream RSV R K T |
| Byte 24 → | 0 0 0 0 1 1 0 0 | DATA DW0 | | |
| Byte 28 → | 1 1 1 0 0 1 1 0 | DATA DW1 | | |
| Byte 32 → | 1 1 0 0 1 1 0 1 | DATA DW2 | | |
| Byte 36 → | 1 1 0 1 1 1 0 1 | DATA DW3 | | |
| Byte 40 → | 0 1 1 1 1 0 0 1 | PCRC | | |
| Byte 44 → | 1 1 0 1 1 0 0 0 | MAC DW0 | | |
| Byte 48 → | 1 1 1 0 1 1 1 0 | MAC DW1 | | |
| Byte 52 → | 1 1 0 1 0 1 0 0 | MAC DW2 | | |

Figure L-4 Example Memory Write IDE TLP (FM) §

L.3 Example NFM IDE TLP With Partial Header Encryption §

The TLP shown in § Figure L-5 , when processed using the Key and IV shown in § Table L-3 , which also includes other relevant inputs and outputs, yields the IDE TLP shown in § Figure L-6 .

Base 6.4 vs Base 6.3

Figure L-5 Example Memory Write TLP with Partial Header Encryption (NFM) §

Table L-3 Inputs and Outputs for Example IDE TLP with Partial Header Encryption (mode 0100b - Address[41:2] Encrypted) (NFM) §

| Input/Outputs variables for AES GCM | Values Byte0 [7:0] byte 1 [15:8] Byte N Key, IV, MAC : Follows AES-GCM Specification with Most significant leftmost |
|---|--|
| <i>Inputs</i> | |
| Key (K) | 8A 95 85 8A 23 1D 41 4A 37 33 F0 E5 7A F6 AE 77 C7 F0 CF BA 75 54 D6 70 83 34 2A 64 2A EF 9D 5D |
| Initialization Vector (IV) | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 |
| A | 92 00 BC 0C 60 00 00 00 04 AB CD 00 FE DC B8 |
| P (including Byte Enables, Address[41:2], and PCRC) | FF A6 1D 95 0C 84 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 9C D1 C4 63 |
| <i>Outputs</i> | |
| C (including encrypted Byte Enables, Address[41:2], and PCRC) | F3 50 AB 19 EE B1 D1 1E C4 B3 FA 2D D4 E1 6E 55 FE 36 E0 2D E0 B8 36 F8 99 8C |
| MAC | 05 F9 2E C3 B3 50 85 45 B1 30 DD A8 |

Figure L-6 Example NFM Memory Write IDE TLP with Partial Header Encryption

L.4 Example FM IDE TLP With Partial Header Encryption §

The TLP shown in § Figure L-7 , when processed using the Key and IV shown in § Table L-4 , which also includes other relevant inputs and outputs, yields the IDE TLP shown in § Figure L-8 .

Base 6.4 vs Base 6.3

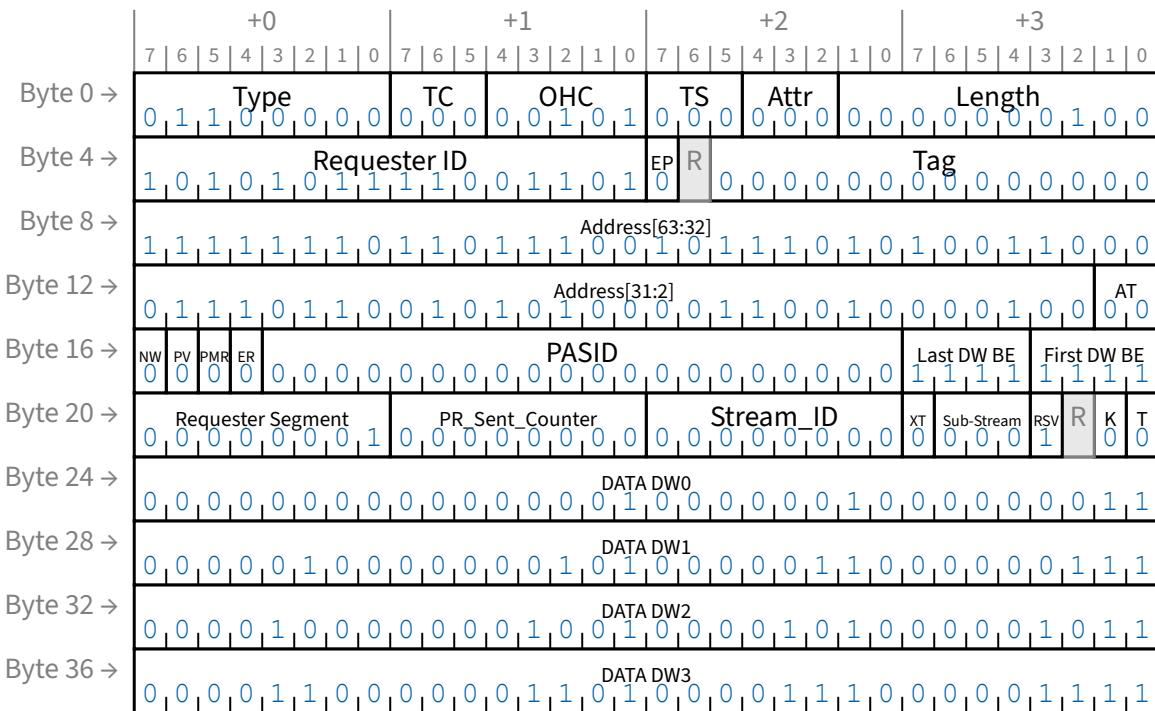


Figure L-7 Example Memory Write TLP with Partial Header Encryption (FM) §

Table L-4 Inputs and Outputs for Example IDE TLP with Partial Header Encryption (mode 0100b - Address[41:2] Encrypted) (FM) §

| | |
|-------------------------------------|--|
| Input/Outputs variables for AES GCM | Values Byte0 [7:0] byte 1 [15:8] Byte N Key, IV, MAC : Follows AES-GCM Specification with Most significant leftmost |
|-------------------------------------|--|

Inputs

| | |
|---|--|
| Key (K) | 8A 95 85 8A 23 1D 41 4A 37 33 F0 E5 7A F6 AE 77 C7 F0 CF BA 75 54 D6 70 83 34 2A 64 2A EF 9D 5D |
| Initialization Vector (IV) | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 |
| A | 60 05 C0 04 AB CD 00 00 FE DC B8 00 00 00 01 00 BC 08 |
| P (including Byte Enables, Address[41:2], and PCRC) | FF A6 1D 95 0C 84 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 9C D1 C4 63 |

Outputs

| | |
|--|--|
| C (including encrypted Byte Enables, Address[41:2] and PCRC) | F3 50 AB 19 EE B1 D1 1E C4 B3 FA 2D D4 E1 6E 55 FE 36 E0 2D E0 B8 36 F8 99 8C |
| MAC | 66 6E 87 A3 36 89 ED 1A C5 17 AD 22 |

```
    "lsbyte": 0, "lsbit": 0, "msbyte": 0, "msbit": 7, "value": ["0", "1", "1", "0", "0", "0", "0"], "name": "Type", "attr": "ro"}, { "lsbyte": 1, "lsbit": 7, "msbyte": 1, "msbit": 5, "name": "TC", "value": ["0", "0", "0"], "attr": "ro"}, { "lsbyte": 1, "lsbit": 4,
```

Base 6.4 vs Base 6.3

```

"msbyte": 1, "msbit": 0, "name": "OHC", "value": ["0", "0", "1", "0", "1"], "attr": "ro"}, {"lsbyte": 2, "lsbit": 7, "msbyte": 2, "msbit": 5, "name": "TS", "value": ["1", "1", "0"], "attr": "ro"}, {"lsbyte": 2, "lsbit": 4, "msbyte": 2, "msbit": 2, "name": "Attr", "value": ["0", "0", "0"], "attr": "ro"}, {"lsbyte": 3, "lsbit": 0, "msbyte": 2, "msbit": 1, "name": "Length", "value": ["0", "0", "0", "0", "0", "0", "1", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 5, "lsbit": 0, "msbyte": 4, "msbit": 7, "name": "Requester ID", "value": ["1", "0", "1", "0", "1", "1", "1", "0", "1", "1", "0", "1", "0", "1"], "attr": "ro"}, {"lsbyte": 6, "lsbit": 7, "msbyte": 6, "msbit": 5, "name": "EP", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 7, "lsbit": 0, "msbyte": 6, "msbit": 6, "name": "Tag", "value": ["0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"], "attr": "ro"}, {"lsbyte": 11, "lsbit": 0, "msbyte": 8, "msbit": 7, "name": "Address[63:32]", "value": ["1", "1", "1", "1", "1", "1", "1", "0", "1", "1", "0", "0", "0", "0", "0", "0", "1", "0", "1", "1", "0", "1", "1", "0", "1", "1", "1", "0", "0", "0", "1", "0", "1", "0", "1"], "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 15, "lsbit": 2, "msbyte": 12, "msbit": 7, "name": "Address[31:2]", "value": ["1", "0", "1", "0", "1", "1", "0", "0", "1", "0", "1", "1", "1", "0", "1", "1", "1", "0", "1", "0", "1", "1", "0", "0", "0", "0", "1"], "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 15, "lsbit": 0, "msbyte": 15, "msbit": 1, "name": "AT", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 7, "msbyte": 16, "msbit": 7, "name": "NW", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 6, "msbyte": 16, "msbit": 6, "name": "PV", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 5, "msbyte": 16, "msbit": 5, "name": "PMR", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 4, "msbyte": 16, "msbit": 4, "name": "ER", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 16, "lsbit": 3, "msbyte": 18, "msbit": 0, "name": "PASID", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 19, "lsbit": 4, "msbyte": 19, "msbit": 7, "name": "Last DW BE", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 19, "lsbit": 0, "msbyte": 19, "msbit": 3, "name": "First DW BE", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 20, "lsbit": 0, "msbyte": 20, "msbit": 7, "name": "Requester Segment", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 21, "lsbit": 0, "msbyte": 21, "msbit": 7, "name": "PR_Sent_Counter", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 22, "lsbit": 0, "msbyte": 22, "msbit": 7, "name": "Stream_ID", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 21, "lsbit": 4, "msbyte": 23, "msbit": 6, "name": "Sub_Stream", "value": "0", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 3, "msbyte": 23, "msbit": 3, "name": "RSV", "value": "1", "addClass": "regFieldVerySmallText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 1, "msbyte": 23, "msbit": 1, "name": "K", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 23, "lsbit": 0, "msbyte": 23, "msbit": 0, "name": "T", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 27, "lsbit": 0, "msbyte": 24, "msbit": 7, "name": "DATA DW0", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 31, "lsbit": 0, "msbyte": 28, "msbit": 7, "name": "DATA DW1", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 35, "lsbit": 0, "msbyte": 32, "msbit": 7, "name": "DATA DW2", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 39, "lsbit": 0, "msbyte": 36, "msbit": 7, "name": "DATA DW3", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 43, "lsbit": 0, "msbyte": 40, "msbit": 7, "name": "PCRC", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 47, "lsbit": 0, "msbyte": 44, "msbit": 7, "name": "MAC DW0", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 51, "lsbit": 0, "msbyte": 48, "msbit": 7, "name": "MAC DW1", "value": "0", "addClass": "regFieldSmallText", "attr": "ro"}, {"lsbyte": 55, "lsbit": 0, "msbyte": 52, "msbit": 7, "name": "MAC DW2", "value": "1", "addClass": "regFieldSmallText", "attr": "ro"} ] ] ]

```

Base 6.4 vs Base 6.3

| | +0 | +1 | +2 | +3 | | | | | |
|-----------|--------------------------------------|------------------------------------|--|-------------------------------|--|---------------------------|------------------------|--------|--------|
| Byte 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 | 7 6 5 4 3 2 1 0 | | | | | |
| Byte 0 → | Type 0 1 1 0 0 0 0 0 | TC 0 0 0 0 0 1 0 1 | OHC 0 1 0 1 1 0 1 0 | TS 1 1 0 0 0 0 0 0 | Attr 0 0 0 0 0 0 0 0 | Length 0 0 0 0 1 0 0 0 | | | |
| Byte 4 → | Requester ID 1 0 1 0 1 0 1 1 | 1 1 1 1 1 1 0 1 1 | Address[63:32] 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 1 0 | EP 1 0 | R | Tag 0 0 0 0 0 0 0 0 | | | |
| Byte 8 → | | | | | | | | | |
| Byte 12 → | | | Address[31:2] 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 | AT 0 0 | | | | | |
| Byte 16 → | NW 0 0 0 0 0 0 0 0 | PV 0 0 0 0 0 0 0 0 | PMR 0 0 0 0 0 0 0 0 | ER 0 0 0 0 0 0 0 0 | PASID 0 0 0 0 0 0 0 0 | Last DW BE 1 1 1 1 | First DW BE 0 0 1 1 | | |
| Byte 20 → | Requester Segment 0 0 0 0 0 0 0 1 | PR_Sent_Counter 0 0 0 0 0 0 0 0 | led 1 0 1 1 1 1 0 0 | Stream_ID 0 0 0 0 0 0 0 0 | XT 0 0 | Sub-Stream 0 0 0 0 1 | RSV R 1 | K 0 | T 0 |
| Byte 24 → | | | | | DATA DW0 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 | | | | |
| Byte 28 → | | | | | DATA DW1 1 1 1 1 1 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 1 0 | | | | |
| Byte 32 → | | | | | DATA DW2 0 1 1 0 1 1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 0 1 1 0 | | | | |
| Byte 36 → | | | | | DATA DW3 1 1 1 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 | | | | |
| Byte 40 → | | | | | PCRC 0 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 | | | | |
| Byte 44 → | | | | | MAC DW0 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 | | | | |
| Byte 48 → | | | | | MAC DW1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 | | | | |
| Byte 52 → | | | | | MAC DW2 1 1 0 0 0 1 0 1 0 0 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 | | | | |

Figure L-8 Example Memory Write IDE TLP with Partial Header Encryption (FM) §

L.5 IDE Test Keys §

The keys defined in this section are intended to be used to simplify IDE validation, for example, by making it easier for verification tools to decrypt and MAC-check TLPs. As illustrated in § Table L-5 , the defined test keys consist of a constant value in the more significant bytes of the key, and an incrementing value contained in the two least significant bytes of the key, where the test key number corresponds to the value of the incrementing portion.

These keys are intended for use in validation and test environments only - it is essential these keys are not used in production environments. It is recommended that verification tools check for the inadvertent use of test keys in production settings, and provide an **↑↓appropriate↑↓appropriate** warning to the user. It is also recommended to check for the use of other key values that have been published as test keys, and for trivial keys such as all 0's.

Table L-5 IDE Test Keys §

| Key Number | Key Value – Byte0 [7:0] byte 1[15:8].....Byte N Key, IV, MAC: Follows AES-GCM Specification with Most significant leftmost |
|------------|--|
| 0 | OB AD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |

| Key Number | Key Value – Byte0 [7:0] byte 1[15:8].....Byte N Key, IV, MAC: Follows AES-GCM Specification with Most significant leftmost |
|------------|---|
| 1 | OB AD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 |
| 2 | OB AD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 |
| ... | OB AD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 xx xx |

Base 6.4 vs Base 6.3

Acknowledgements

The following persons were instrumental in the development of this specification:²³⁰

| Name | Affiliation |
|----------------------------|------------------------------|
| Chamath Abhayagunawardhana | Intel Corporation |
| Ryan Abraham | Oracle Corporation |
| Shiva Aditham | Intel Corporation |
| Hong Ahn | Xilinx |
| Shay Aisman | Mellanox Technologies, Ltd. |
| Jasmin Ajanovic | Intel Corporation |
| Katsutoshi Akagi | NEC Corporation |
| Lee Albion | Intel Corporation |
| Praveen Alexander | Microchip |
| Joe Allen | Tektronix, Inc. |
| Michael W. Altmann | Intel Corporation |
| Taha Amiralli | Advanced Micro Devices, Inc. |
| Boon Ang | VMware Corporation |
| Keng Dar Ang | Intel Corporation |
| Ishwar Agarwal | Microsoft Corporation |
| Peter Arnoldy | Cadence Design Systems, Inc. |
| Sujith Arramreddy | ServerWorks, Inc. |
| Antonio Asaro | Advanced Micro Devices, Inc. |
| Ahmad Atamli | NVIDIA Corporation |
| Yuval Avnon | Marvell Semiconductor, Inc. |
| Jay Avula | ServerWorks, Inc. |
| Pervez Aziz | NVIDIA Corporation |
| Tony Bacchillone | Synopsys, Inc. |
| Brian Baldwin | Synopsys, Inc. |
| Jasper Balraj | Intel Corporation |
| Nat Barbiero | Advanced Micro Devices, Inc. |

²³⁰ Company affiliation listed is at the time of specification contributions.

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-------------------|------------------------------|
| Phillip Barnes | Hewlett-Packard Company |
| Jim Bartenslager | Credo Semiconductor, Inc. |
| Suparna Behera | LSI Logic Corporation |
| Joseph A. Bennett | Intel Corporation |
| Richard Bell | Advanced Micro Devices, Inc. |
| Stuart Berke | Hewlett-Packard Company |
| Mike Beyers | Keysight |
| Harish Bharadwaj | LSI Logic Corporation |
| Ajay V. Bhatt | Intel Corporation |
| Gaurav Bhide | Advanced Micro Devices, Inc. |
| Harmeet Bhugra | IDT Corporation |
| Christiaan Bil | Intel Corporation |
| Bill Bissonette | Intel Corporation |
| Cass Blodget | Intel Corporation |
| Jeffrey D. Bloom | Intel Corporation |
| Mahesh Bohra | IBM Corporation |
| Naveen Bohra | Intel Corporation |
| Karl Bois | Hewlett Packard Enterprise |
| Austin Bolen | Dell Inc. |
| Jerry Bolen | Intel Corporation |
| Michele Bologna | Synopsys, Inc. |
| Wil de Bont | National Instruments |
| David Bouse | Tektronix, Inc. |
| Hicham Bouzekri | ST-Ericsson |
| Gavin Bowlby | Marvell Semiconductor, Inc |
| Suri Brahmaroutu | Dell Inc. |
| Chris Brokke | VTM Group |
| Dave Brown | Integrated Device Technology |
| Tory Brown | Intel Corporation |
| Mike Brownell | Intel Corporation |
| Bilal Butt | Keysight |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|------------------------|-------------------------------|
| Bala Cadambi | Intel Corporation |
| John Calvin | VTM |
| John Calvin | Keysight |
| Jun Cao | Synopsys, Inc. |
| Gord Caruk | Advanced Micro Devices, Inc. |
| Paul Cassidy | Synopsys, Inc. |
| Alex Chadwick | Arm Limited |
| Craig Chaiken | Dell Technologies |
| James Chapple | Intel Corporation |
| Kabitha Chaturvedula | LSI Logic Corporation |
| Yogesh Chaudhary | Mentor Graphics |
| Elene Chobanyan | Hewlett-Packard Enterprise |
| Santanu Chaudhuri | Intel Corporation |
| Rajinder Cheema | LSI Logic Corporation |
| Albert Chen | LSI Logic Corporation |
| Chih-Cheh Chen | Intel Corporation |
| Jian Chen | Parade |
| Qunwei Chen | Advanced Micro Devices, Inc. |
| Tony Chen | Marvell Semiconductor, Inc |
| Jonathan Cheung | Advanced Micro Devices, Inc. |
| Yorick Cho | Advanced Micro Devices, Inc. |
| Michael Choi | Samsung Electronics Co., Ltd. |
| Phillip Chopp | Synopsys, Inc. |
| Dickson Chow | Advanced Micro Devices, Inc. |
| Saad Muhammad Chughtai | Tektronix, Inc. |
| Gene Chui | IDT Corporation |
| Kee Shik Chung | Marvell Semiconductor, Inc. |
| Shawn Clayton | Emulex Corporation |
| Shaun Clem | Synopsys, Inc. |
| Mark Clements | IBM Corporation |
| Debra Cohen | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|---------------------|--------------------------------|
| Eric Combs | LSI Logic Corporation |
| Brad Congdon | Intel Corporation |
| Gustavo Contreras | Intel Corporation |
| Mike Converse | IDT Corporation |
| Justin Coppin | Hewlett-Packard Company |
| Joe Cowan | Hewlett-Packard Enterprise |
| Carrie Cox | IBM Corporation |
| Ethan Crain | Advanced Micro Devices, Inc. |
| H. Clay Cranford | IBM Corporation |
| Kenneth C. Creta | Intel Corporation |
| Pamela Cristo | Marvell Semiconductor, Inc |
| Sanjay Dabral | Intel Corporation |
| Eric Dahlen | Intel Corporation |
| Tugrul Daim | Intel Corporation |
| Ron Dammann | Intel Corporation |
| Sumit Das | Texas Instruments Incorporated |
| Debendra Das Sharma | Intel Corporation |
| Nicole Daugherty | Intel Corporation |
| Glen Dearth | Advanced Micro Devices, Inc. |
| Eric DeHaemer | Intel Corporation |
| Fei Deng | Intel Corporation |
| Dan DeSetto | Intel Corporation |
| Karishma Dhruv | LSI Logic Corporation |
| Gary Dick | Cadence Design Systems, Inc. |
| Robert Dickson | Oracle Corporation |
| Jin Ding | Synopsys, Inc. |
| Bob Divivier | IDT Corporation |
| Hormoz Djahanshahi | Microsemi Corporation |
| Daniel Dreps | IBM Corporation |
| Ken Drottar | Intel Corporation |
| Dave Dunning | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|---------------------------|------------------------------|
| Rick Eads | Keysight |
| Gregory L. Ebert | Intel Corporation |
| Samer El-Haj-Mahmoud | Arm Limited |
| Yuval Elad | Arm Limited |
| Imtinan Elahi | NVIDIA Corporation |
| Yaron Elboim | Intel Corporation |
| Bassam Elkhoury | Intel Corporation |
| Salem Emara | Advanced Micro Devices, Inc. |
| Mike Engbretson | Tektronix, Inc. |
| Raul Enriquez | Intel Corporation |
| Jonathon Evans | NVIDIA Corporation |
| Matt Evans | Arm Limited |
| Ohad Falik | Intel Corporation |
| David Fair | Intel Corporation |
| Zhineng Fan | Amphenol Corp |
| Blaise Fanning | Intel Corporation |
| John Feehrer | Oracle Corporation |
| Will Felten | Synopsys, Inc. |
| Wes Ficken | GLOBALFOUNDRIES Inc. |
| Joshua Filliater | LSI Logic Corporation |
| Michael Fleischer-Reumann | Agilent Technologies, Inc. |
| Pelle Fornberg | Intel Corporation |
| Jeff Fose | Emulex Corporation |
| Jim Foster | IBM Corporation |
| Mike Foxcroft | Advanced Micro Devices, Inc. |
| Douglas Freimuth | IBM Corporation |
| Daniel S. Froelich | Intel Corporation |
| SivaPrasad Gadey | Intel Corporation |
| Mathieu Gagnon | Cadence Design Systems, Inc. |
| Yoni Galezer | Mellanox Technologies, Ltd. |
| Eric Geisler | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|---------------------|------------------------------|
| Gordon Getty | Teledyne LeCroy |
| Gregory Geyfman | Intel Corporation |
| John Geldman | Kioxia |
| Subhankar Ghose | Tektronix, Inc |
| Stefano Giacconi | Intel Corporation |
| Charles Giefer | Arm Limited |
| Dave Gines | Keysight |
| Steve Glaser | NVIDIA Corporation |
| Thorsten Goetzlmann | Keysight |
| Marc A. Goldschmidt | Intel Corporation |
| Dean Gonzales | Advanced Micro Devices, Inc. |
| Alan Goodrum | Hewlett-Packard Company |
| Robert Gough | Ampere Computing |
| Lucian Gozu | Neterion Corporation |
| Ilya Granovsky | IBM Corporation |
| Brien Gray | Intel Corporation |
| Greg Green | Marvell Semiconductor, Inc. |
| Richard Greene | Intel Corporation |
| Stephen Greenwood | ATI Technologies Inc. |
| Michael J. Greger | Intel Corporation |
| Buck Gremel | Intel Corporation |
| Andrew Gruber | Advanced Micro Devices, Inc. |
| George Gruber | Qualcomm |
| Vijay Gudur | LSI Logic Corporation |
| Mark Guenther | Tektronix, Inc. |
| Dale Gulick | Advanced Micro Devices, Inc. |
| Jong-ru Guo | Intel Corporation |
| Liping Guo | Marvell Semiconductor, Inc. |
| Mickey Gutman | Intel Corporation |
| Clifford D. Hall | Intel Corporation |
| Stephen H. Hall | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-------------------------|--------------------------------|
| Joseph Hamel | IBM Corporation |
| Moiz Haq | Advanced Micro Devices, Inc. |
| Ken Haren | Intel Corporation |
| David Harriman | Ampere Computing |
| Hiromitsu Hashimoto | NEC Corporation |
| George R. Hayek | Intel Corporation |
| Mike Yun He | Intel Corporation |
| Wenmu He | Texas Instruments Incorporated |
| Matt Hendrick | Intel Corporation |
| Hamish Hendry | Cadence Design Systems, Inc. |
| Brett Henning | Broadcom |
| Roque Arcudia Hernandez | Cadence Design Systems, Inc. |
| Michael Herz | Blackberry |
| Bent Hessen-Schmidt | Tektronix, Inc. |
| Hanh Hoang | Intel Corporation |
| Joseph Kho Boon Hock | Intel Corporation |
| Michael Hopgood | NVIDIA Corporation |
| Dorcas Hsia | NVIDIA Corporation |
| James Huang | Advanced Micro Devices, Inc. |
| Robert Huang | NVIDIA Corporation |
| Yifan Huang | Amphenol-FCI |
| Kamlesh Hujwant | Advanced Micro Devices, Inc. |
| Mark Hummel | Advanced Micro Devices, Inc. |
| Mikal Hunsaker | Intel Corporation |
| Scott Huss | Cadence Design Systems, Inc. |
| Joaquin Ibanez | Synopsys, Inc. |
| Yasser Ibrahim | Microsoft Corporation |
| Franko Itay | Intel Corporation |
| Carl Jackson | Hewlett-Packard Company |
| David R. Jackson | Intel Corporation |
| Praveen Jain | NVIDIA Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|----------------------------|--------------------------------|
| Martin James | Cadence Design Systems, Inc. |
| Duane January | Intel Corporation |
| Ihab Jaser | Microchip |
| James E. Jaussi | Intel Corporation |
| J, Jebaselvi | Primesoc Technologies |
| Mike Jenkins | LSI Logic Corporation |
| Peter Jenkins | Marvell Semiconductor, Inc |
| Hong Jiang | Intel Corporation |
| Viek Joshi | Intel Corporation |
| Dave Kaffine | Oracle Corporation |
| David Kahmayhew | Oracle Corporation |
| Ravi Kammaje | LSI Logic Corporation |
| Girish Karanam | LSI Logic Corporation |
| Kapil Karkra | Intel Corporation |
| Navnit Kashyap | Cadence Design Systems, Inc. |
| Chad Kendall | Broadcom Inc. |
| Patrick Kennedy | Intel Corporation |
| Shashitheren Al A Kerisnan | Intel Corporation |
| Umair Khan | Intel Corporation |
| Mukund Kharti | Dell Computer Corporation |
| Gyanaranjan Khuntia | Mentor Graphics |
| David Kimble | Texas Instruments Incorporated |
| Lavi Koch | NVIDIA Corporation |
| Mohammad Kolbehdari | Intel Corporation |
| Abhimanyu Kolla | Intel Corporation |
| Ganesh Kondapuram | Intel Corporation |
| Kwok Kong | IDT Corporation |
| Michael Krause | Hewlett-Packard Enterprise |
| Gopi Krishnamurthy | Cadence Design Systems, Inc. |
| Sreenivas Krishnan | NVIDIA Corporation |
| Kumaran Krishnasamy | Broadcom Inc. |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|--------------------|------------------------------|
| Steve Krooswyk | Samtec |
| Masahiro Kudo | Socionext Inc. |
| Manjari Kulkarni | Intel Corporation |
| Yashodhan Kulkarni | Keysight |
| Akhilesh Kumar | Intel Corporation |
| Deepak Kumar | Synopsys, Inc. |
| Mohan J. Kumar | Intel Corporation |
| M Satish Kumar | Cadence Design Systems, Inc. |
| Santosh Kumar | SK Hynix |
| Richard Kunze | Intel Corporation |
| Hugh Kurth | Oracle Corporation |
| Seh Kwa | Intel Corporation |
| Ricky Lai | Hewlett Packard Enterprise |
| Eric N. Lais | IBM Corporation |
| Bill Lam | eTopus Technology Inc. |
| Sunny Lam | Intel Corporation |
| Tim Lambert | Dell Computer |
| Dave Landsman | Western Digital |
| Jon Lange | Microsoft Corporation |
| Brian Langendorf | NVIDIA Corporation |
| Raymond R. Law | Intel Corporation |
| Dror Lazar | Intel Corporation |
| Brian L'Ecuyer | Agilent Technologies, Inc. |
| Beomtek Lee | Intel Corporation |
| Clifford D. Lee | Intel Corporation |
| David M. Lee | Intel Corporation |
| Edward Lee | Advanced Micro Devices, Inc. |
| Justin Lee | Credo Semiconductor, Inc. |
| Hye Su Lee | NVIDIA Corporation |
| Ming Chew Lee | Intel Corporation |
| Moshe Leibowitz | IBM Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-------------------|-----------------------------------|
| Adrian Leuciuc | Cadence Design Systems, Inc. |
| Tony L. Lewis | Intel Corporation |
| Jingbo Li | Intel Corporation |
| Mike Peng Li | Intel Corporation |
| Paul Li | Pericom Semiconductor Corporation |
| Stephen Li | Texas Instruments Incorporated |
| Ying Li | NVIDIA Corporation |
| Tao Liang | Intel Corporation |
| Andrew K. Lillie | Intel Corporation |
| Akan Lin | MediaTek |
| Wendy Liu | Advanced Micro Devices, Inc. |
| Ivan Lobachev | Intel Corporation |
| Hooi Kar Loo | Intel Corporation |
| Jeff Lukanc | IDT Corporation |
| Jeffrey Lu | IBM Corporation |
| Betty Luk | Advanced Micro Devices, Inc. |
| Tianchen Luo | Credo Semiconductor, Inc. |
| Ngoc Luu | Advanced Micro Devices, Inc. |
| Kenneth Ma | Broadcom Inc. |
| Stephen Ma | Advanced Micro Devices, Inc. |
| Zorik Machulsky | IBM Corporation |
| Mallik Mahalingam | VMware, Inc. |
| Surabhi Mahata | Marvell Semiconductor, Inc. |
| Kevin Main | Texas Instruments Incorporated |
| Steven Makow | IBM Corporation |
| Anubhav Mangla | Synopsys, Inc. |
| Tony Mangefeste | Microsoft Corporation |
| Steve Manning | Advanced Micro Devices, Inc. |
| Gold Mao | VIA Technologies, Inc. |
| Jarek Marczewski | Advanced Micro Devices, Inc. |
| Mark Marlett | LSI Logic Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|--------------------|--|
| John Maroney | Micron Technology |
| Alberto Martinez | Intel Corporation |
| Andrew Martwick | Intel Corporation |
| Anish Mathew | Cadence Design Systems, Inc. |
| Don Matthews | Advanced Micro Devices, Inc. |
| Bruce Mathewson | Arm Limited |
| Paul Mattos | GLOBALFOUNDRIES Inc. |
| Robert A. Mayer | Intel Corporation |
| David Mayhew | (Stargen, Inc.) Advanced Micro Devices, Inc. |
| Mohiuddin Mazumder | Intel Corporation |
| Joe McCann | Synopsys, Inc. |
| David McTavish | Advanced Micro Devices, Inc. |
| Larry McMillan | Western Digital |
| Alexander McVay | Independent |
| Bob McVay | Parade Technologies, Inc. |
| Mehdi Mechaik | Cadence Design Systems, Inc. |
| Mehdi M. Mechaik | NVIDIA Corporation |
| Pranav H. Mehta | Intel Corporation |
| Vishal Mehta | NVIDIA Corporation |
| Richard Mellitz | Intel Corporation |
| Adi Menachem | Valens Semiconductor, Ltd. |
| Cindy Merkin | Dell Computer Corporation |
| Slobodan Milijevic | Microsemi Corporation |
| Dennis Miller | Intel Corporation |
| Jason Miller | Oracle Corporation |
| Robert J. Miller | Intel Corporation |
| Michael Mirmak | Intel Corporation |
| Mitch Miskoski | NVIDIA Corporation |
| Suneel Mitbander | Intel Corporation |
| Mohammad Mobin | NVIDIA Corporation |
| Kerul Modi | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-----------------------------------|----------------------------------|
| Daniel Moertl | IBM Corporation |
| Ravi Mohanavelu | Intel Corporation |
| Se-jung Moon | Intel Corporation |
| Lee Mohrmann | National Instruments Corporation |
| Carlos Lizalde Moreno | Intel Corporation |
| Puga Nathal Moises | Intel Corporation |
| Richard Moore | Cavium Inc. |
| Wayne Moore | Intel Corporation |
| Douglas R. Moran | Intel Corporation |
| Terry Morris | Hewlett-Packard Enterprise |
| Jeff C. Morriss | Intel Corporation |
| Linna Mu | Avago Technologies |
| Sridhar Muthrasanallur | Intel Corporation |
| Suresh Babu M. V. | LSI Logic Corporation |
| Ezra Baruch | Intel Corporation |
| Gautam V. Naik | LSI Logic Corporation |
| Mohan K. Nair | Intel Corporation |
| Mukund Narasimhan | Intel Corporation |
| Vasudev Uttharabhalli NarayanaDev | Granite River Labs |
| Alon Naveh | Intel Corporation |
| Ramin Neshati | Intel Corporation |
| Surena Neshvad | Intel Corporation |
| Andy Ng | IDT Corporation |
| Uyen Nguyen | Cadence Design Systems, Inc. |
| Manisha Nilange | Intel Corporation |
| Paul Nitza | Marvell Semiconductor, Inc |
| Marianne Nourzad | Intel Corporation |
| Hajime Nozaki | NEC Corporation |
| Kugao Ohuchi | NEC Corporation |
| Paul Olarig | Samsung Electronics Co., Ltd. |
| Vitor Oliveira | Synopsys, Inc. |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-----------------------|------------------------------|
| Olufemi Oluwafemi | Intel Corporation |
| Takuya Omura | Synopsys, Inc. |
| Peter Onufryk | Integrated Device Technology |
| Mike Osborn | Advanced Micro Devices, Inc. |
| Jake Oshins | Microsoft Corporation |
| Randy Ott | Intel Corporation |
| Jonathan Owen | Advanced Micro Devices, Inc. |
| Ali Oztaskin | Intel Corporation |
| David Pabisz | Oracle Corporation |
| Raveendra Pai G | Cadence Design Systems, Inc. |
| Shreeram Palghat | Intel Corporation |
| Jim Panian | Qualcomm |
| Chetan A Paragaonkar | Cadence Design Systems, Inc. |
| Petar Pepeljugsoski | IBM Corporation |
| Marc Pegolotti | ServerWorks, Inc. |
| Henry Peng | Intel Corporation |
| Akshay Pethe | Intel Corporation |
| Chris Pettey | NextIO, Inc. |
| Tien Pham | Hewlett Packard Enterprise |
| Lu-vong Phan | Intel Corporation |
| Lorenzo Pieralisi | Arm Limited |
| Tony Pierce | Microsoft Corporation |
| Daniel Sern Hong Phan | Intel Corporation |
| Edmund Poh | Molex, Inc. |
| Harshit Poladia | Intel Corporation |
| Prasanth R Posam | Intel Corporation |
| Edoardo Prete | Advanced Micro Devices, Inc. |
| Jim Prijic | Intel Corporation |
| Dave Puffer | Intel Corporation |
| Duane Quiet | Intel Corporation |
| Jeffrey D. Rabe | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-------------------|-------------------------------|
| Andy Raffman | Microsoft Corporation |
| Kianoush Rahbar | Intel Corporation |
| Guru Rajamani | Intel Corporation |
| Ramesh Raman | LSI Logic Corporation |
| Adee Ran | Intel Corporation |
| Thanu Rangarajan | Arm Limited |
| Todd Rasmus | IBM Corporation |
| Renato Recio | IBM Corporation |
| Ramakrishna Reddy | LSI Logic Corporation |
| Jack Regula | PLX Technology, Inc. |
| Dick Reohr | Intel Corporation |
| Curtis Ridgeway | Synopsys, Inc. |
| Dwight Riley | Hewlett-Packard Enterprise |
| Yoav Rozenbert | Mellanox Technologies, Ltd. |
| Chris Runhaar | NVIDIA Corporation |
| Rajanataraj S. | LSI Logic Corporation |
| Biswaranjan Sahoo | Samsung Electronics Co., Ltd. |
| Devang Sachdev | NVIDIA Corporation |
| Ahmed Sada | Synopsys, Inc. |
| Gene Saghi | Broadcom Inc. |
| Toshiaki Sakai | Socionext Inc |
| Varun Sampath | NVIDIA Corporation |
| Rajesh Sankaran | Intel Corporation |
| Bill Sauber | Dell Computer Corporation |
| Joseph Schachner | Teledyne LeCroy |
| Mike Schaecher | Oracle Corporation |
| Joe Schaefer | Intel Corporation |
| Michael Scheid | Broadcom Inc. |
| Daren Schmidt | Intel Corporation |
| Mark Schmisseur | Intel Corporation |
| Richard Schober | NVIDIA Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|----------------------------|------------------------------|
| Zale Schoenborn | Intel Corporation |
| Rick Schuckle | Dell Computer Corporation |
| Richard Schumacher | Hewlett-Packard Enterprise |
| Jeremiah Schwartz | Intel Corporation |
| Tudor Secasius | Intel Corporation |
| Kazunori Seki | Synopsys, Inc. |
| Oren Sela | Mellanox Technologies, Ltd. |
| Kevin Senohrabek | Advanced Micro Devices, Inc. |
| Kalev Sepp | Tektronix, Inc. |
| Glen Sescila | Dell Technologies |
| Kossi Sessou | Intel Corporation |
| Prashant Sethi | Intel Corporation |
| Ankur Shah | Intel Corporation |
| Vedvyas Shanbhogue | Intel Corporation |
| Vasudevan Shanmugasundaram | Intel Corporation |
| Richard Shannon | Ampere Computing |
| Wesley Shao | Oracle Corporation |
| Prateek Sharma | Micron Technology |
| Charlie Shaver | Hewlett-Packard Company |
| Yoni Shternhell | Western Digital |
| Kyle P Sheahan | Intel Corporation |
| Robert Sheldon | Oracle Corporation |
| John Sheplock | IBM Corporation |
| Wenjun Shi | NVIDIA Corporation |
| Milton Shih | Oracle Corporation |
| Mark Shillingburg | Agilent Technologies |
| Oren Shirak | Mellanox Technologies, Ltd. |
| Sarvesh Shrivastava | Qualcomm |
| Bill Simms | NVIDIA Corporation |
| Vinita Singhal | NVIDIA Corporation |
| Siddharth Shukla | Cadence Design Systems, Inc. |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|--------------------|---------------------------------|
| Shyamal Sivanandan | Cadence Design Systems, Inc. |
| Rob Sleigh | Keysight |
| Dan Slocombe | Cadence Design Systems, Inc. |
| Brian Small | Northwest Logic, Inc. |
| George Smith | Microsoft Corporation |
| Shamnad SN | LSI Logic Corporation |
| Rick Sodke | PMC-Sierra |
| Gary Solomon | Intel Corporation |
| Richard Solomon | Synopsys, Inc. |
| Gao Song | IDT Corporation |
| Brad Sonksen | Marvell Semiconductor, Inc |
| Walter Soto | Broadcom Inc. |
| Fulvio Spagna | Intel Corporation |
| Jason Squire | Molex, Inc. |
| Anupriya Sriramulu | Advanced Micro Devices, Inc. |
| Patrick Stabile | Oracle Corporation |
| Sean O. Stalley | Intel Corporation |
| Stan Stanski | IBM Corporation |
| Karsten Stangel | Intel Corporation |
| Hermann Stehling | Bitifeye Digital Test Solutions |
| John T. Stonick | Synopsys, Inc. |
| Paul Suhler | Kioxia |
| Phil Sun | Credo Semiconductor, Inc. |
| Eugene Sushansky | Granite River Labs |
| Andrew Swaine | Arm Limited |
| Ron Swartz | Intel Corporation |
| Tim Symons | PMC-Sierra |
| Miki Takahashi | NEC Corporation |
| Gerry Talbot | Advanced Micro Devices, Inc. |
| Anthony Tam | Advanced Micro Devices, Inc. |
| Kan Tan | Tektronix, Inc. |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|----------------------|------------------------------|
| Wanru Tao | Oracle Corporation |
| Mark Taylor | NVIDIA Corporation |
| Matthew Tedone | LSI Logic Corporation |
| Grigori Temkine | Advanced Micro Devices, Inc. |
| Peter Teng | NEC Corporation |
| Bruce A. Tennant | Intel Corporation |
| Poh Thiam Teoh | Intel Corporation |
| Larry Tesdall | Marvell Semiconductor, Inc |
| Tessil Thomas | Arm Limited |
| Andrew Thornton | Microsoft Corporation |
| Steven Thurber | IBM Corporation |
| Mike Tobin | Intel Corporation |
| Ben Toby | Hewlett Packard Enterprise |
| Duke Tran | Broadcom Inc. |
| Tan V. Tran | Intel Corporation |
| Alok Tripathi | Intel Corporation |
| William Tsu | NVIDIA Corporation |
| Alexander Umansky | Huawei Technologies Co. |
| Chris Van Beek | Intel Corporation |
| Arie van der Hoeven | Microsoft Corporation |
| Andrew Vargas | Intel Corporation |
| Robertson Velez | ATI Technologies Inc. |
| Archana Vasudevan | LSI Logic Corporation |
| Kiran Velicheti | Intel Corporation |
| Balaji Vembu | Intel Corporation |
| Gary Verdun | Dell Computer Corporation |
| Sushil Verghese | Broadcom Inc. |
| Divya Vijayaraghavan | Altera Corporation |
| Ravindra Viswanath | LSI Logic Corporation |
| Pete D. Vogt | Intel Corporation |
| Andrew Volk | Intel Corporation |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|--------------------|------------------------------|
| Mahesh Wagh | Intel Corporation |
| Clint Walker | Intel Corporation |
| Davis Walker | Microsoft Corporation |
| Hui Wang | IDT Corporation |
| Kai A. Wang | Intel Corporation |
| Leo Warmuth | NXP Semiconductors |
| Dan Wartski | Intel Corporation |
| Neil Webb | Cadence Design Systems, Inc. |
| Eric Wehage | Huawei Technologies Co. |
| Dong Wei | Arm Limited |
| Ron Weimer | QLogic Corporation |
| Amir Wiener | Intel Corporation |
| Marc Wells | Intel Corporation |
| Rob Wessel | Hewlett-Packard Company |
| Stephen F. Whalley | Intel Corporation |
| Bryan White | Intel Corporation |
| Paul Whittemore | Oracle Corporation |
| Amir Wiener | Intel Corporation |
| Timothy Wig | Intel Corporation |
| Craig Wiley | Parade |
| Jim Williams | Emulex Corporation |
| Theodore L. Willke | Intel Corporation |
| Dawn Wood | Intel Corporation |
| John Wood | Emulex Corporation |
| David Woodral | QLogic Corporation |
| David Wooten | Microsoft Corporation |
| Zhi Wong | Altera Corporation |
| David Wormus | LSI Logic Corporation |
| David Wu | Broadcom Inc. |
| Hsinho Wu | Intel Corporation |
| William Wu | Broadcom Inc. |

Base 6.4 vs Base 6.3

| Name | Affiliation |
|-----------------|--------------------------------|
| Zuoguo Wu | Intel Corporation |
| Kai Xiao | Intel Corporation |
| Ping Xiong | Credo Semiconductor, Inc. |
| Liu Xin | IDT Corporation |
| Dan Yaklin | Texas Instruments Incorporated |
| Howard Yan | Intel Corporation |
| Jane Yan | Oracle Corporation |
| Al Yanes | IBM Corporation |
| John Y Yang | Intel Corporation |
| Koon Fatt Yap | NVIDIA Corporation |
| Danny Ybarra | Western Digital |
| Gin Yee | Advanced Micro Devices, Inc. |
| Ahmed Younis | Xilinx, Inc. |
| Chi-Ho Yue | NVIDIA Corporation |
| Wayne Yun | Advanced Micro Devices, Inc. |
| Dave Zenz | Dell Computer Corporation |
| Shubing Zhai | IDT Corporation |
| Wei Zhang | Broadcom Inc. |
| Bo Zhang | Intel Corporation |
| Guoqing Zhang | Cadence Design Systems, Inc. |
| Liwei Zhao | Intel Corporation |
| Xiaoshu Zhao | Intel Corporation |
| Yang Zhixin | Huawei Technologies Co. |
| Charlie Ziegler | Dell Computer Corporation |
| Kevin Ziegler | Tektronix, Inc. |
| Aibing Zhou | Fungible |
| Yaping Zhou | NVIDIA Corporation |

